LIBRARY COPY

The Sinclair Survivalist HANDBOOK

ZX81/TS1000

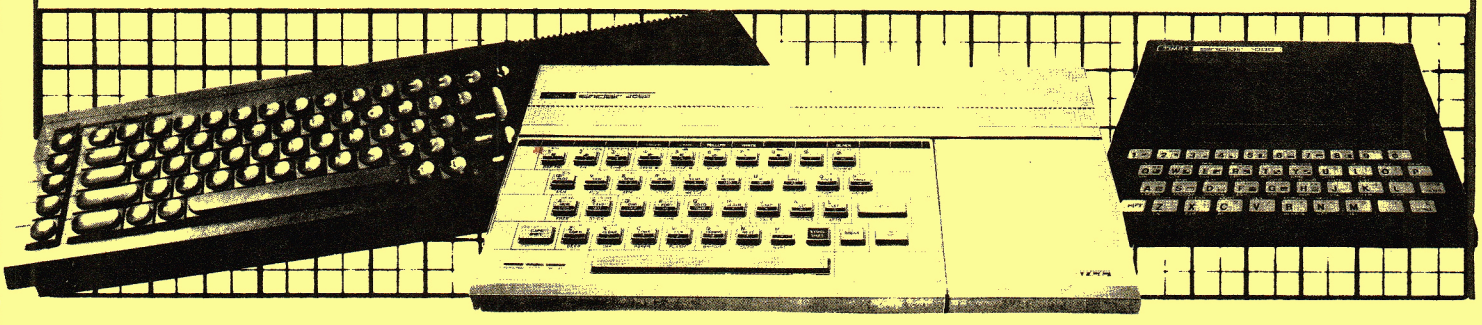
TS1500

TS2068

SPECTRUM

QL

Published by Time Designs Magazine



The Sinclair Survivalist HANDBOOK

Published by Time Designs Magazine

THE SINCLAIR SURVIVAL HANDBOOK

(c) Copyright 1988 by Time Designs Magazine Co.

All rights reserved. Reproduction of this book in whole or in part by any means, electronic, mechanical, photocopying, or otherwise, without written permission from the publisher, is prohibited by law.

Notice: Contributors to "The Sinclair Survival Handbook" are independent of Time Designs Magazine Co., and opinions which are expressed in the contents of this publication are not necessarily those of the management staff.

Time Designs Magazine Co. will not be held liable for any damage or consequences resulting from instructions, assertion of fact, review of products or companies, provided in the contents of this publication.

It is recommended that anyone attempting to modify their computer or peripheral equipment, or construct an electronic device, should seek further assistance from idividuals who are more knowledgeable on the subject.

The Sinclair Survival Guide was conceived, designed and edited by Tim Woods. Further editing by Stephanie Woods.

ISBN #

Printed in the United States of America

Time Designs Magazine Company
29722 Hult Rd., Colton, OR 97017

"THE SINCLAIR SURVIVAL HANDBOOK" is dedicated to all those who contributed material to the project and granted permission for its use herein, and also to anyone who wishes to see our great little microcomputers survive.

Table of Contents

Introduction.....II

ZX81/TS1000/TS1500 Section

New Keyboards For Old..... 1
Deleting A Block Of Lines In BASIC..... 7
80 Column Printer Utility.....10
"Real Computing Power" For Your ZX/TS.....11
TS1000 and TS1500 I/O Compatibility.....12
Strategic Football.....14

TS2068/Spectrum Section

Build A Spectrum Romswitch In Your TS2068.....17
TS2068 Automatic Emergency Power Supply.....18
Fix That 2068 Space Bar.....19
League Scheduler.....20
Polynomial Curve Fit Program.....25
Dice II.....30
Simple Snap Grid For Paul Bingham's UEDG.....31
Draw Poker.....32
Black Jack.....35
Multiplication.....38
Utilities For OS-64
Metamorphosis.....39
Dexter Et Sinister.....40
Hooray For HISOFT.....42
Max Headroom.....46
Light Pens.....47
Fairlight.....48
Spitfire 40.....50

Sinclair QL Section

QL Qorner.....51
Quill Update For The Sinclair QL.....55
Text Adventure Game Example.....57
Multi-Quill Copier.....58
The Heart Of The Sinclair QL.....59
More On Qooling The QL.....60

INTRODUCTION

Perhaps the title of this new publication may at first be a bit deceiving. A "survival guide" for Sinclair computer owners? Visions of a manual which describes which fungi is non-poisonous, or how to build a shelter out of scrub brush, may come to mind. But just like its outdoors counterpart, our survival guide dispenses important information. For the key to survival is knowledge.

After four years of publishing TIME DESIGNS, a bi-monthly magazine, our file of contributed articles and programs did "runneth over". Its difficult to try and pack everything in one magazine especially when news and advertisements are essential.

The Sinclair Survival Guide was our best solution to disperse as much information as possible to our readership. We felt that it was wrong to keep excellent contributions like these filed away. And if this idea works well, and we are faced with a similar "problem" down the line, who knows, we may publish a "Survival Guide Part 2".

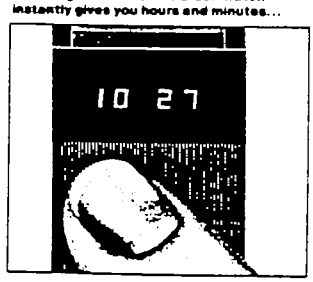
If you are like us, you have probably been a Sinclair fan for some time. Maybe you even owned the BLACK WATCH depicted in the old advertisement depicted below. But more than likely you have a more recent microchip invention such as a ZX81, QL, or a Timex product like the TS1500 or TS2068. Whatever the computer model, I am sure that you have discovered the surprising power and uniqueness that is built into the computer's design.

Our uncle, the inventor, Sir Clive Sinclair is not finished tinkering with technology. In a recent interview, he painted future visions of servant personal robot droids, a telephone that goes everywhere with you, and 20 megabytes of RAM on a four inch slice of silicon.

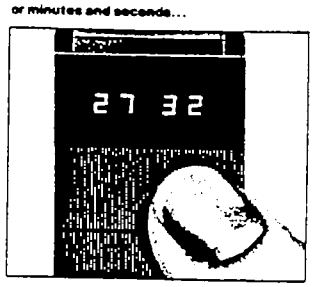
What does this all have to do with a survival guide for microcomputers? We really aren't orphans or survivalists, but merely on the cutting edge of technology!

- Tim Woods

Touch and tell!
Touching the case of the Black Watch instantly gives you hours and minutes...



or minutes and seconds...



No fiddly, fussy knobs or buttons.

Try the Black Watch on a 10-day money-back guarantee.

The Black Watch is exclusive, though not expensive. The design is very advanced, and we believe that you should have every opportunity to see it and wear it without commitment.

You are therefore invited to try it on a 10-day money-back guarantee. It's not everybody's watch, and we shall fully understand if you eventually decide you would like something more conventional.

To try the Black Watch, simply fill in the order form below and send it with your check, or charge card details. We'll send you a Black Watch direct

To: Sinclair Radionics Inc., 375 Park Avenue, New York, N.Y. 10022.

I am interested in trying the Black Watch

☐ Please send me....(qty) at $49.95 with wear-tested matt-black strap.

☐ Please send me....(qty) at $54.95 with matt-black stainless steel bracelet.

I enclose check/m.o. for $_____

☐ Please charge my credit card account

Credit Card_____

Account #_____

Expires_____

Signature_____

N.Y. residents please add sales tax.

Name (please print)_____

Address_____

City_____ State_____ Zip_____

Sinclair Radionics Inc.
375 Park Avenue,
New York,
N.Y. 10022

sinclair

NEW KEYBOARDS FOR OLD

by Ken Malpas

There is now an abundance of both Texas Instrument keyboards and Timex Sinclair computers on the surplus market, and their similarity in wiring makes them near perfect mates for each other. All one need do is a little rewiring and the old fragile membrane keyboard can be replaced by conventional keys, the kind the "real guys" use.

First, let me say that my knowledge of things electrical is very limited, and that when I started this enterprise I knew nothing of how a keyboard worked, or what the inside of my computer looked like. I had two sets of instructions to go by which agreed with each other in certain ways and disagreed in certain ways. In the article you are reading, I have attempted to consolidate the best parts of each, and I also mean to mention a few points which they both neglected to explain.

Beneath each key on the TI keyboard are two terminals ("pins") which become connected when the key is depressed. This action connects "vertical" and "horizontal" lines as shown in figure 1. Because of this grid, this is called a "matrix keyboard." Each key is known to the computer by a unique combination of two electrical impulses, one along a "vertical" and the other along a "horizontal" line. When these messages intersect in the computer, they convey the signal designated on the key. These "vertical" and "horizontal" lines are not to be confused with what seems "vertical" or "horizontal" when viewing the QWERTYUIOP typewriter keyboard from the normal aspect. Learn to view the keys according to this diagram.

Since the wiring patterns for keyboards used by Texas Instrument vary according to manufacturer, it is suggested that the re-wirer (you) familiarize himself with the vertical and horizontal terminals beneath each key. This can be found by turning the keyboard over and identifying the external leads for the keyboard (shown in figures 2 and 7) as either vertical or horizontal, and following their lines around to the various pins. In doing this, the re-wirer should be aware that lines can seem mysteriously to end in unnumbered locations, only to have their reappearance in nearby unnumbered locations. These are *sub rosa* connections, tying several groups of pins along the line.

I made a photocopy (figure 3) of my own keyboard and marked vertical terminals blue and the horizontal ones red.

When reading lines and pins, it is necessary to avoid confusion. Line 8 is different from key number 8, and so on (key number 8 is not even *on* line number 8).

From viewing figure 2, it will be noticed that the TI, with 15 external leads, has two leads not connected to the Timex, which has only 13. Although the two keyboard formats have eight vertical lines, the TI has two extra horizontal ones (figure 2). One, line number 12, connects to most of the operator keys (SHIFT, etc.), and number 6 goes to ALPHA LOCK. These lines are not used in the Timex. In figure 1 you can see that the remaining rows and columns would be the same as Timex if a few keys were moved about, and that is what we are about to do.

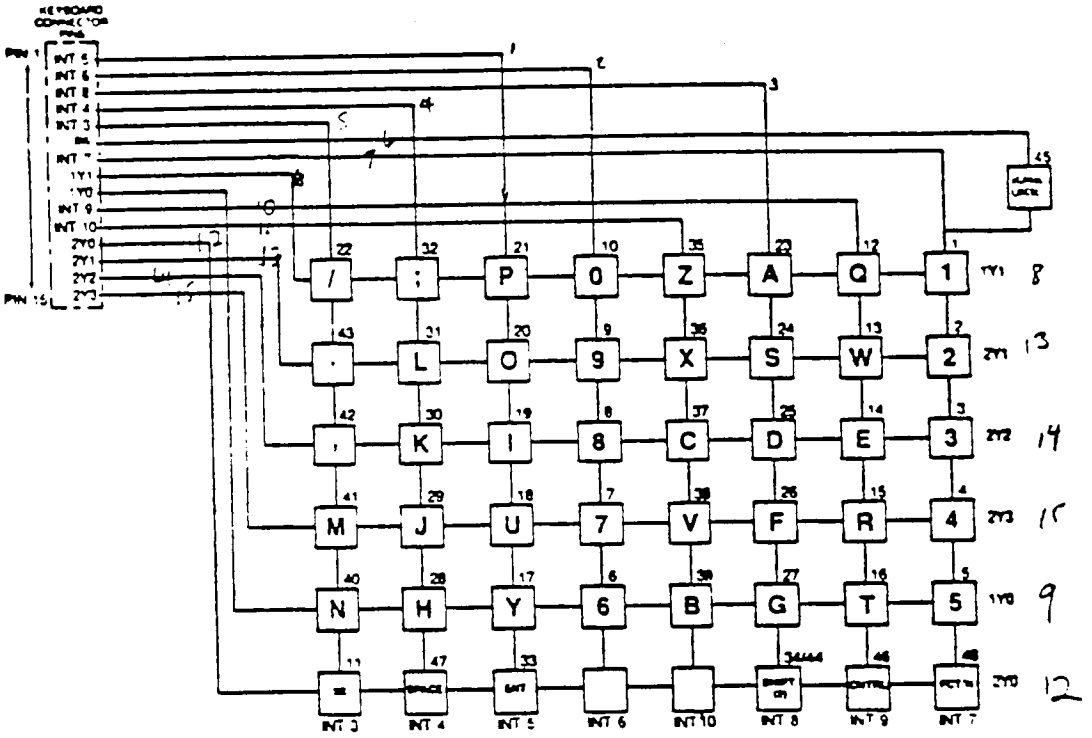
Figure 4 shows the layout as we wish it to be arranged. Each key you see on this chart will have vertical lines connecting it with the one above and below it, and each key will have horizontal lines connecting it with the one to the left and right of it. The shaded keys are those which have been subject to relocation, and all others remain as they are. Acquaint yourself with the original positions of these entities before relocation (code numbers will remain constant with key symbol), and with their new positions. Z, X, C, and V move down in respect to their horizontal neighbors. B moves over beneath N; N and M move higher; . and , are consolidated into one key, and . becomes a SPACE key. SHIFT is brought into the new order by connecting unused line 12 to vertical 11, converting formerly horizontal 12 to a vertical line continuous with 11, and tying the other (formerly vertical, now horizontal) pin to line 8.

In order to show how these migrations are to be effected, let us use keys M and N as examples (figure 5). Originally, M is connected horizontally with J. This connection must be broken on both sides of the horizontal pin for M, and connected with the next horizontal line above it. Because of the break in horizontal line 15, where M used to be, it is necessary to re-establish that connection with a jumper wire between any two points in that line which have suffered alienation, preferably two whose pins are located nearby each other on the keyboard bottom, so the cable will be short. This should be done after all other modifications have been made, as it will then be clearer as to the best sites for these wires. Most of the lines to be altered are horizontal; only the line between B and V gets severed and B connected to N vertically.

Figure 7 shows how I have done mine. The repetitious succession of nine connections along the near bottom (32-43, 31-42, etc.) are the displacement of keys from the horizontals; those long ones near the top are the "jumper wires." Other wires shown on this diagram are optional finesses which will be explained further on.

Now we are ready to begin the operation. Following are the precise manipulations. I cannot account for whether in your keyboard the pin in question has one or two lines going to it, so I have elected to use the expression "totally isolating the pin from the line." In practice, it means severing one or both lines to that pin. If only one line goes to the pin, the other pins on that line will not have to be re-connected, and no jumper wire will be needed.

.(43) will yield SPACE by totally isolating the horizontal .(43) pin from all other pins on line 13, and connecting it to some conveniently located pin on line 8. If this involves severing the line on two sides of the pin, a jumper wire will be needed to bypass the break on line 13. The SPACE BAR can be brought into effect by breaking lines to both of its pins and connecting them to lines 8 and 5.



KEYBOARD SCHEMATIC

FIGURE 1

TI-COMPUTER CONNECTIONS
(AS SEEN FROM TOP)

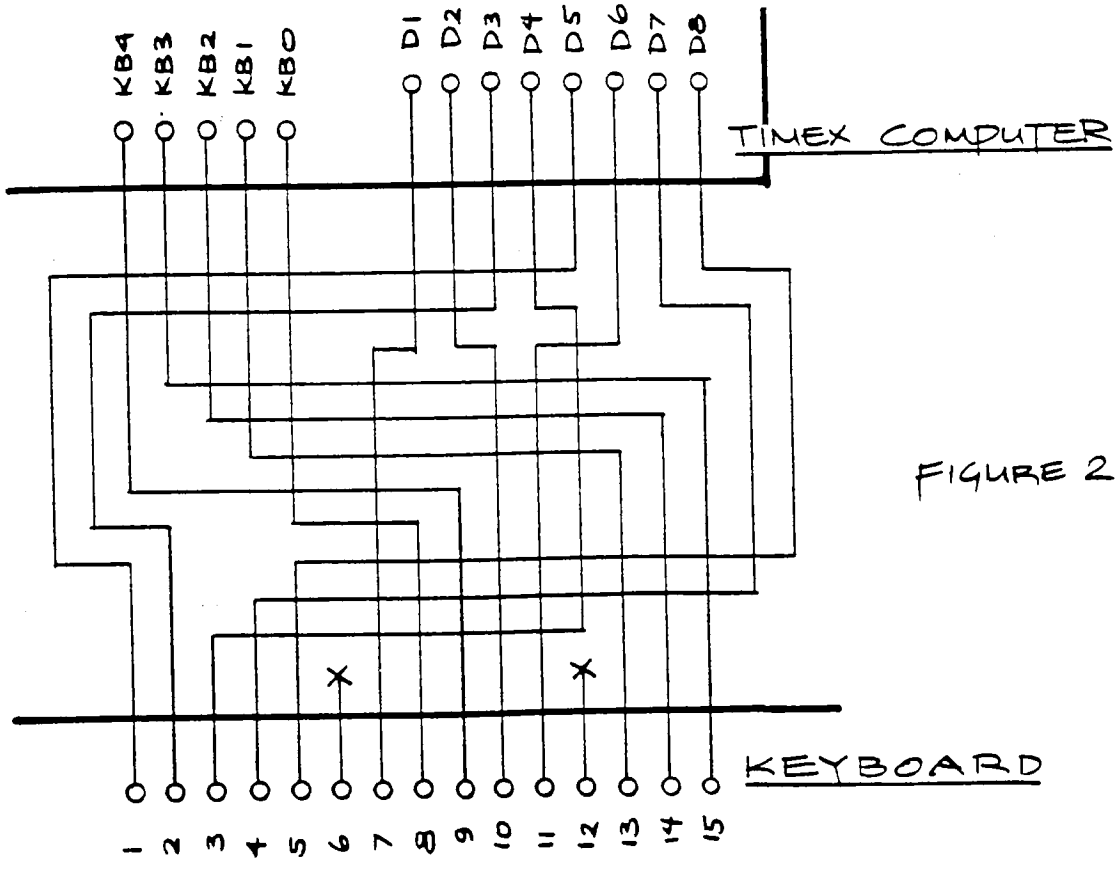


FIGURE 2

ORIGINAL TI LAYOUT

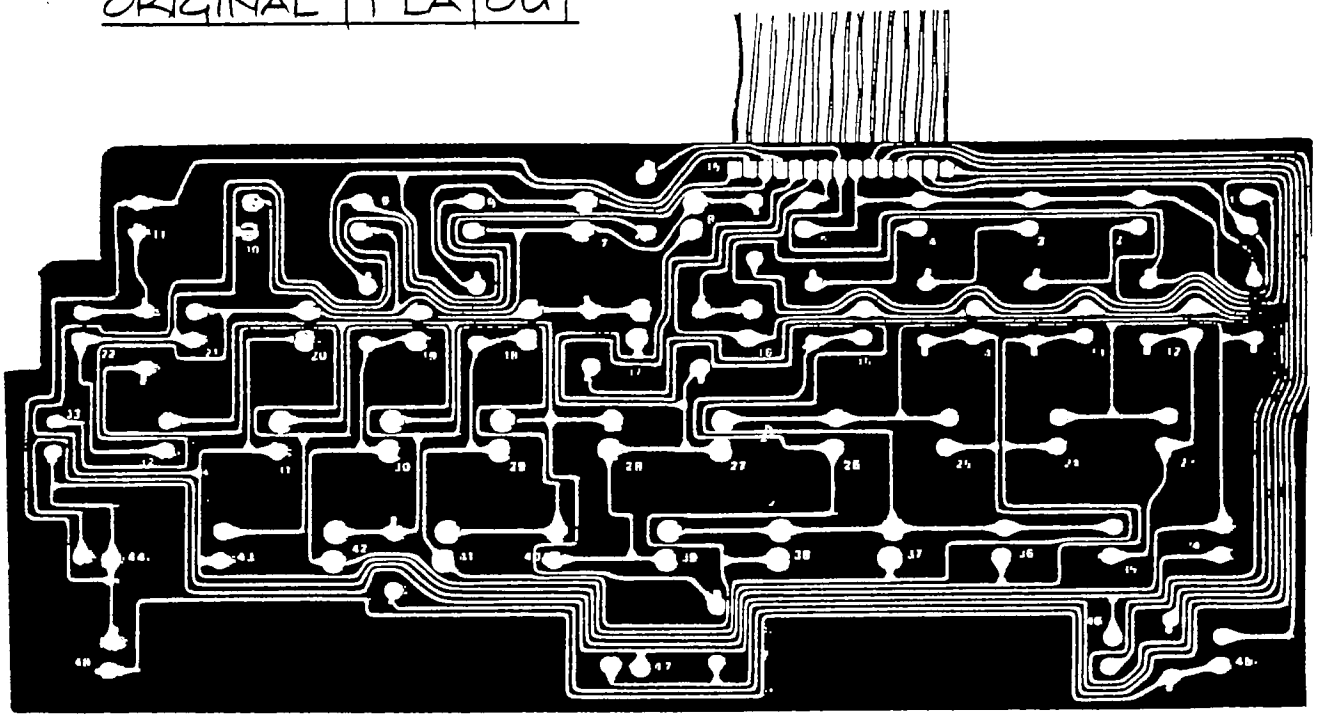


FIGURE 3

,(42) will yield "." by totally isolating the horizontal ,(42) pin from all other pins on line 14, and connecting it to some conveniently located pin on line 13. If this involves severing the line on two sides of the pin, a jumper wire will be needed to bypass the break on line 14.

M(41) will yield "M" by totally isolating the horizontal M(41) pin from all other pins on line 15, and connecting it to some conveniently located pin on line 14. If this involves severing the line on two sides of the pin, a jumper wire will be needed to bypass the break on line 15.

N(40) will yield "N" by totally isolating the horizontal N(40) pin from all other pins on line 9, and connecting it to some conveniently located pin on line 15. If this involves severing the line on two sides of the pin, a jumper wire will be needed to bypass the break on line 9.

B(39) will yield "B" by totally isolating the vertical B(39) pin from all other pins on line 11, and connecting it to any conveniently located pin on line 5. If this involves severing the line on two sides of the pin, a jumper wire will be needed to bypass the break on line 11.

V(38) will yield "V" by totally isolating the horizontal V(38) pin from all other pins on line 15, and connecting it to any conveniently located pin on line 9. If this involves severing the line on two sides of the pin, a jumper wire will be needed to bypass the break on line 15.

C(37) will yield "C" by totally isolating the horizontal C(37) pin from all other pins on line 14, and connecting it to any conveniently located pin on line 15. If this involves severing the line on two sides of the pin, a jumper wire will be needed to bypass the break on line 14.

X(36) will yield "X" by totally isolating the horizontal X(36) pin from all other pins on line 13, and connecting it to any conveniently located pin on line 14. If this involves severing the line on two sides of the pin, a jumper wire will be needed to bypass the break on line 13.

Z(35) will yield "Z" by totally isolating the horizontal Z(35) pin from all other pins on line 8, and connecting it to any conveniently located pin on line 13. If this involves severing the line on two sides of the pin, a jumper wire will be needed to bypass the break on line 8.

SHIFT(34/44) can be activated by doing two things: Link unused line 12 to line 11 by connecting the erstwhile "horizontal" pin at 34 to any conveniently located pin on line 11. There is no need to sever line 12. Next, cut line 3 between A-S-D-F-G and SHIFT(34). Connect the newly isolated SHIFT pin to any conveniently located pin on horizontal line 8. Only key 34 need be rewired, as key 44 is in parallel with it.

Make an electric series with a battery, a voltage meter, and wire leads to ascertain that the right pins are now connected to the right lines. If you connect this series to horizontal pin of key 43, for instance, will get a reading when the other end of the meter's the cable is at the external lead number 8. Do this for the horizontal and vertical lines of each key.

Before making the final link of the keyboard to the computer, the more ambitious re-wirer may wish to make the following refinements on his keyboard, provided that the keys on his keyboard are positioned, as mine are, to be thus taken advantage of.

1. An extra ENTER key, to the right of the usual one. Sever both lines to key 33 and connect to 32.

2. You may want to de-activate the SPACE BAR occasionally to avoid hitting it accidentally during a program and causing the program to BREAK. Since there is no need for a SHIFT LOCK with the Timex computer, your ALPHA LOCK key can be used as a switch to activate or de-activate the SPACE BAR. It can be wired as illustrated in figure 7; the schematic is shown in figure 6.

5	4	1	2	11	3	10	7	
43 SPACE	32 EN	21 P	10 0	34 SHIFT	23 A	12 Q	1 1	8
42 *	31 L	20 O	9 9	33 Z	24 S	13 W	2 2	13
41 M	30 K	19 I	8 8	32 X	25 D	14 E	3 3	14
40 N	29 J	18 U	7 7	31 C	26 F	15 R	4 4	15
39 B	28 H	17 Y	6 6	30 V	27 G	16 T	5 5	9

FIGURE 4

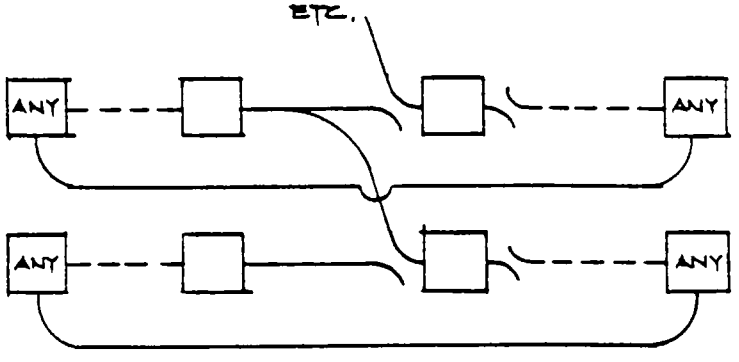


FIGURE 5

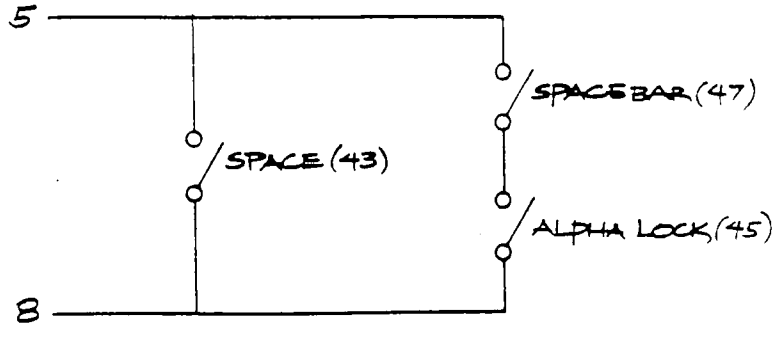


FIGURE 6

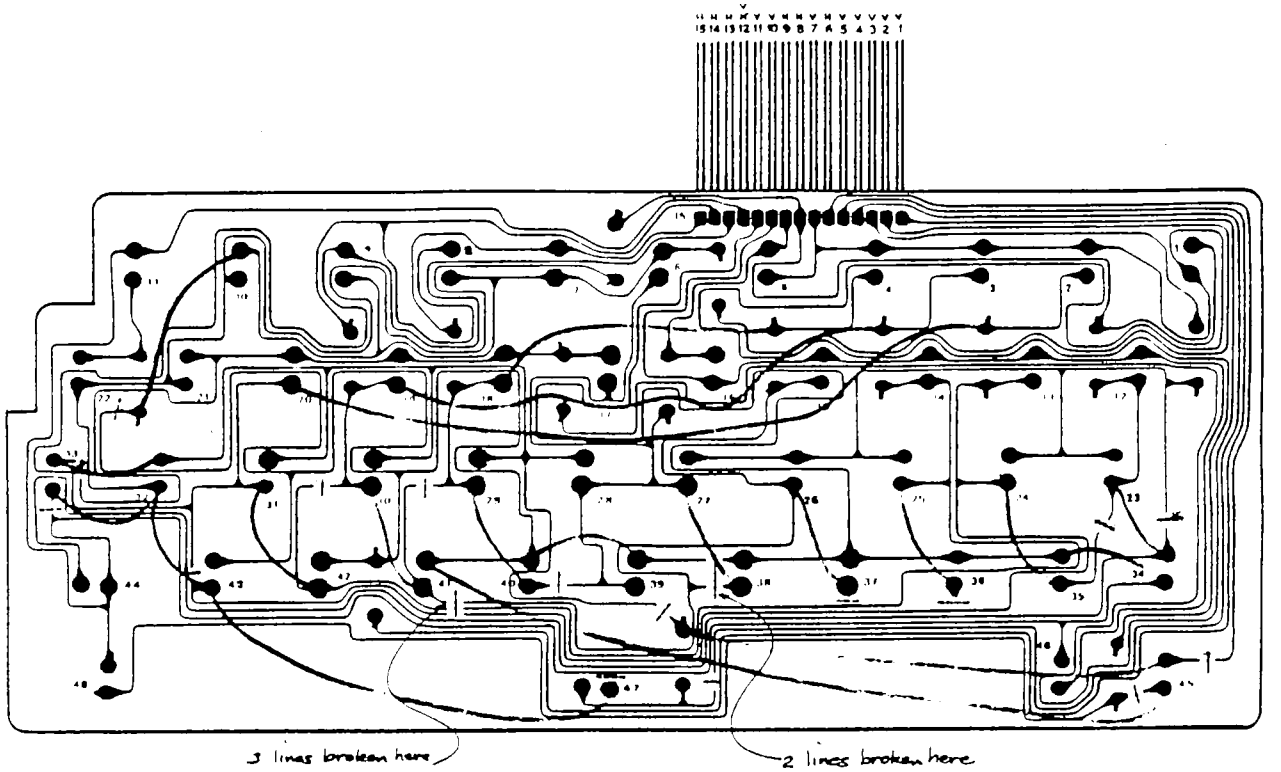


FIGURE 7

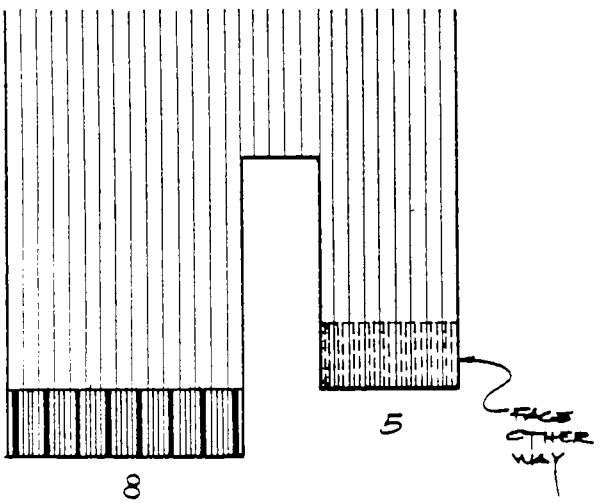
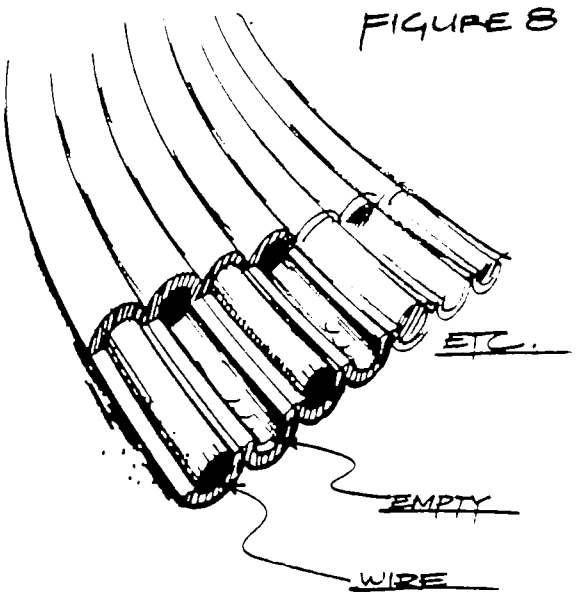


FIGURE 8



3. A press-only-once DELETE key can be made from keys 11 and 22 by gluing a plastic shim underneath the two keys so that when key 11 is depressed, key 22 goes down with it (only key 22 need be glued to the shim). The wiring for this involves isolating the vertical pins for keys 11 and 22, and the establishment of their connection to any convenient pin on line 2. See figure 7.

4. Without any further re-wiring, FCTN(48) will yield you an extra EDIT key.

Now that the keyboard proper is re-wired, it can be connected to the computer in the manner designated by figure 2. As you can see, the wires reach across one another and change positions. They can be soldered to the TI keyboard leads, or a fancier disconnect apparatus can be employed.

There are five screws which hold the upper and lower halves of the Timex together, some of which can be seen by removing the rubber pods on which the computer stands. Remove the five screws and open the case. It can be seen that the keyboard is connected by membrane ribbons set, by pressure only, into two units containing electrical contacts. The locations of the terminals on these two units can also be observed in figure 2.

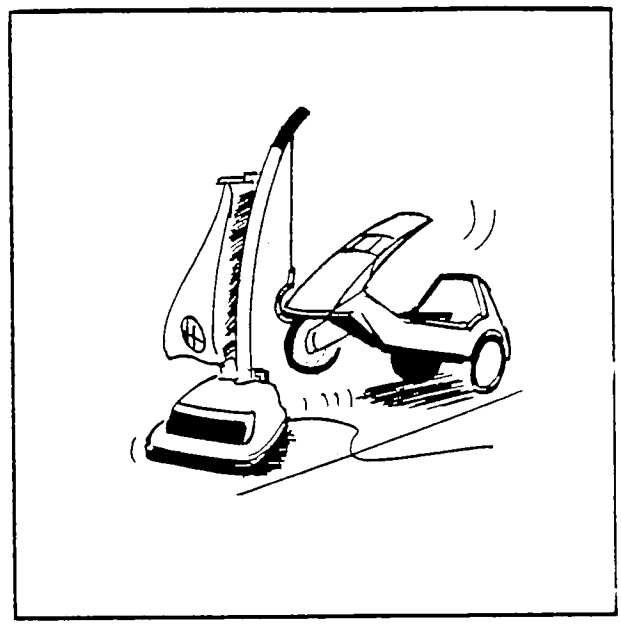
Take these ribbons out and fold them back out of the way. These two units will also admit conventional wire ribbons. The insulation must be stripped off *one side* of the wire ribbon so as to bare the wire on a single side only, and the entire ribbon end pressed firmly into the receiving unit. See figure 8. Notice that the five contacts of the "horizontal line" unit are oriented in an opposite direction to the eight contacts of the "vertical line" unit, requiring that the ends for vertical be exposed differently from those for horizontal. Secondly, notice that the horizontal unit is closer to the edge of the motherboard than the vertical; you will want to make

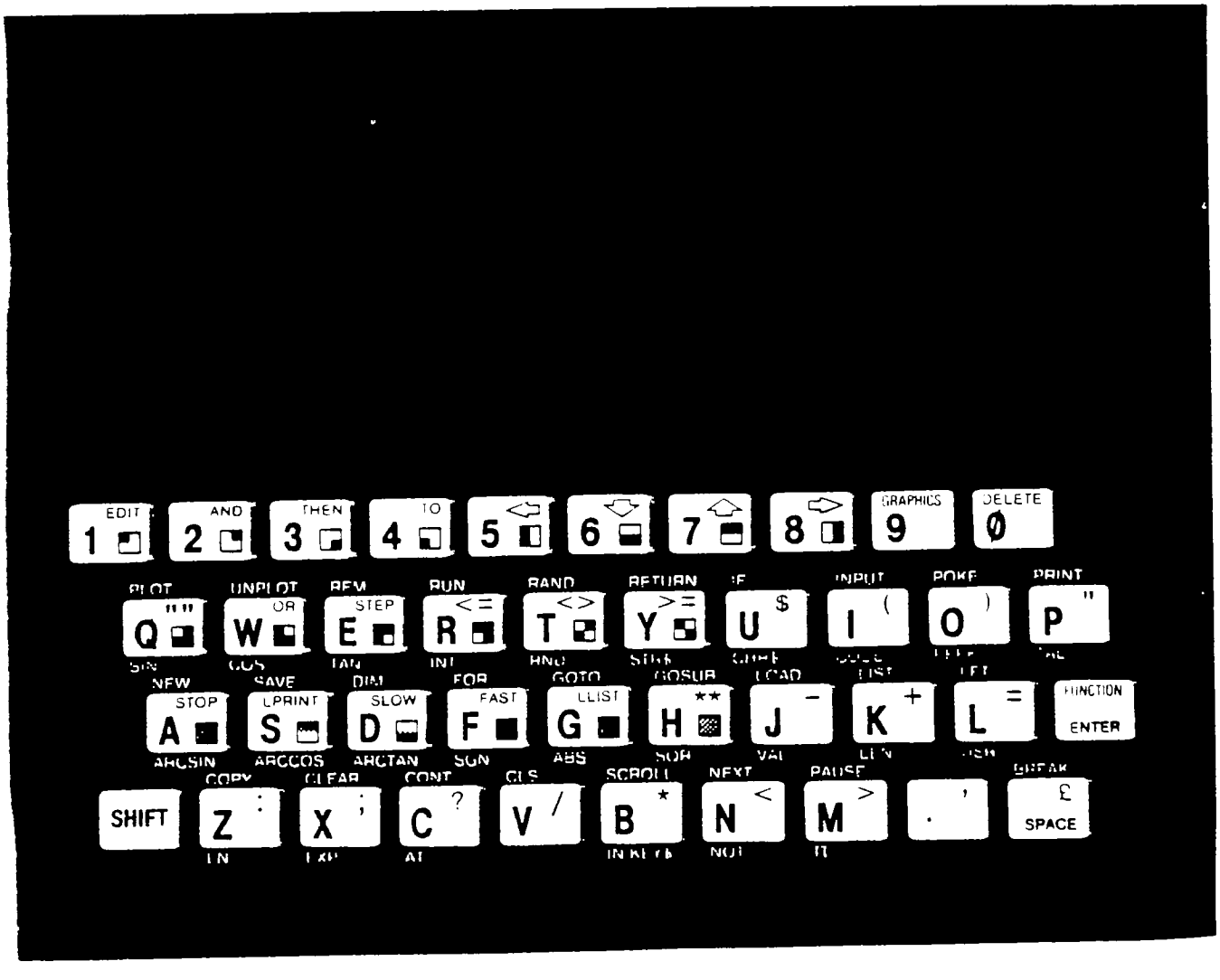
the length of ribbon going to one unit slightly longer than the other. Finally, take note of the fact that the electrical contact points in the computer connector units are set at distances of every other wire in a wire ribbon, so you will need to use only every other wire in the wire ribbon if you plan to insert it in the connections.

The end of any of the wires in the wire ribbon should be soldered ("tinned") in order to make its many strands into one filament. You do not want them to "frizz out" or cross over to a neighboring connection when inserted into the connector unit. When you are sure every wire is in position, firmly push the ends of the ribbon into the connector units.

The case of the computer can be shut, with the wire ribbon extruding from the front of it, but you might have to file away at the case slightly.

It has been suggested by one writer that the membrane to the old keyboard be cut up and each key from the membrane glued to the keys of the new keyboard. This has the advantage of allowing the operator to see readily the key symbols, but I recommend keeping a drawing or photocopy of the keyboard for that purpose. That way, you won't be burning your bridges after yourself.





DELETING A BLOCK OF LINES IN BASIC (and Machine Code)

by Gale Henslee

In the process of developing programs for the ZX81/TS1000, I have occasionally found a need to delete a large block of lines from a program. With the ZX/TS built-in editor, this is a tiresome chore involving repeatedly typing in blank program lines to delete the unwanted code. Most editor programs incorporate a function which automatically deletes a block of lines. I set out to determine whether the block delete function could be programmed in Sinclair BASIC without resorting to machine code, and ended up with the program which is the subject of this article.

Of course, the program execution is very slow and the program could be made much shorter by converting to machine code. This would also increase the execution speed dramatically, and I will provide an example for demonstration purposes. However, it is not necessary to convert to machine code for the program to work.

In order to delete a program line, the Sinclair editor locates the beginning and ending addresses for that line and then moves the next line and all the following code down to the address previously occupied by the deleted line. We will follow the same process, simply locating the beginning and ending addresses for a block of lines instead of for a single line. We will also have to change the addresses stored in three important system variables in order to recover the display and the variables intact.

Figure 1 illustrates the necessary elements for decoding a line in Sinclair BASIC in order to locate the line number, line length, beginning address, and ending address, using Line 1 as an example. Enter Line 1 on your computer as:

1 REM 00000000000000 (1 REM followed by 14 zeros and ENTER). The fourteen zeros will be needed for storage locations later.

Each BASIC line ends with ENTER, code 118, and we could find the end of each line by searching each byte to see if it is an ENTER. However, this is not necessary, or very efficient, since we can use the two bytes following the line number to calculate where the end of the present line is and the address of the next lines start. The line number, which occupies 2 bytes, is followed by a two byte number containing the length of the text following it plus one for ENTER.

Decoding line numbers is straightforward, but note that in contrast to most other two byte numbers in the Z80, the line number is stored with its more significant byte (MSB) first. Each byte contains the hexadecimal representation of a number from 0 to 255. If you PEEK the first two addresses of a line and get 10 and 20 respectively, the corresponding line number is calculated as (10*256) plus 20=2580. If you let s = the starting address of the line, then the line number is stored in locations S and S+1.

If you continue PEEKing the next two locations, S+2 and S+3, you can calculate the line length as LENGTH=PEEK(S+2)++256*PEEK(S+3)+4. The four bytes at the end of the formula are the four bytes S, S+1, S+2, and S+3. If you add LENGTH+S, you get the starting address of the next line. LENGTH+S-1 is the address of the last byte in the current line.

You now have the information to locate the beginning and ending addresses for a single line or a block of lines, so we can now start the program.

Enter lines 1 to 30 as shown in Listing 1. To test this segment type RUN and enter beginning and ending line numbers (use 900 and 950 for testing). You should then see the line numbers echoed in the top right corner of the screen and be prompted to verify the numbers. Type "N" and ENTER, and the line numbers should disappear, prompting you to enter them again. Enter the line numbers again and type "Y", or just ENTER and you will be asked for a starting address. Input "0" and ENTER and the screen will briefly flash "START=16509" before going into fast mode. Try inputting a number other than 16509 and make sure that number is echoed correctly.

When section 1 works properly, enter lines 32 thru 72. These lines use the previously entered values of S, B, and E, to locate the beginning and ending addresses of desired lines. They print these addresses, and give you a chance to decide if you still want to delete them. If you enter "N", then the program stops. (This can be used to locate the storage address of a block of code for some other purpose.) If you enter "Y", the program continues and, in line 72, calculates D, which is the displacement of the following addresses required to delete the block of lines. The input and output prompts that create the utility screen plus all the REM statements that are used to identify blocks of code in the program take up over 500 bytes of storage. Except for the input prompts for the beginning and ending line numbers, none of them are really necessary, and these lines can be removed if storage requirements are critical. In fact, the final version of the program does not contain them. They are given in Listing 1 in order to aid in learning the function of each block of code.

Figure 1. Sinclair BASIC Program Line Organization

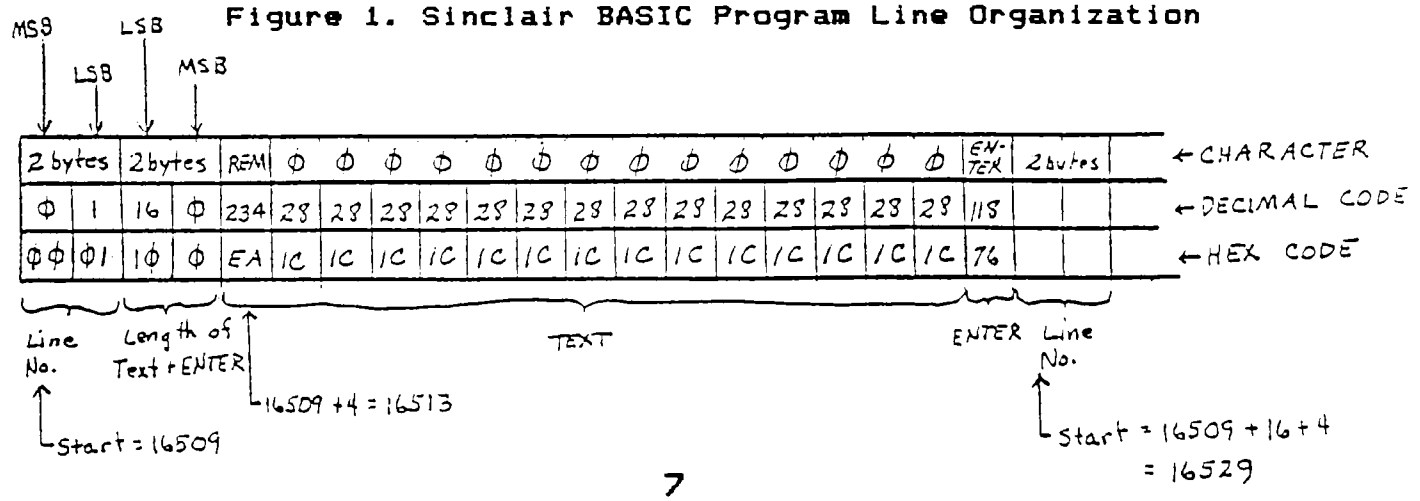
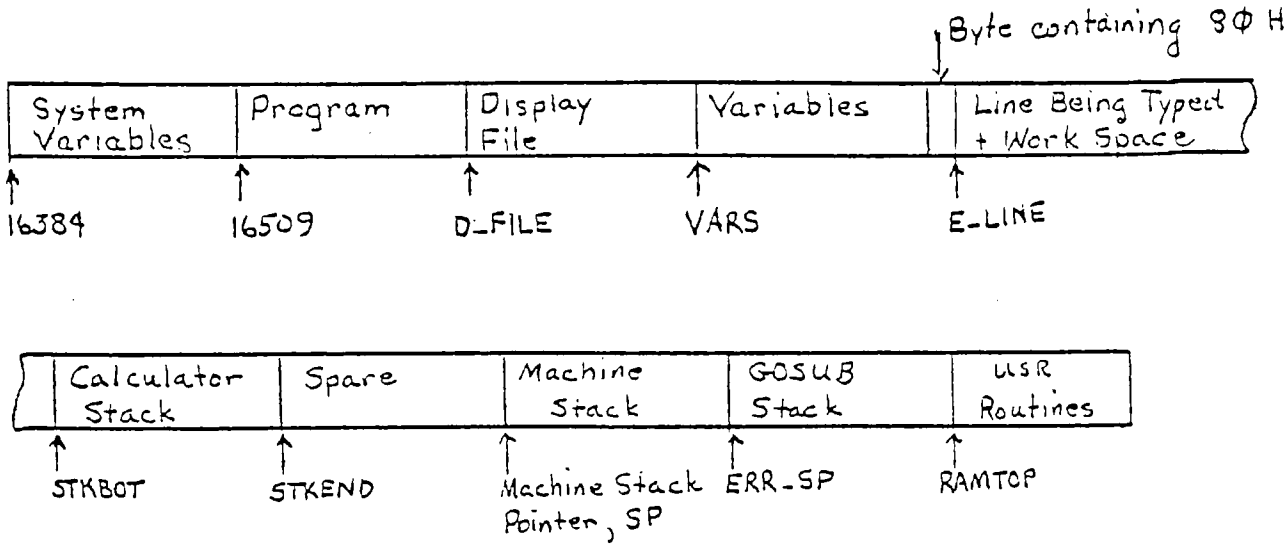


Figure 2. Organization of Program Storage in the TS1000



Now enter lines 74 to 120. This portion of the program is concerned with storing the addresses of the system variables which must be restored after the block move. It also stores the address of the first byte after the code being deleted, the address to which it is to be moved, the displacement, and the pointer for the next byte to be copied. These numbers are stored in locations 16514 thru 16527, which were created in the REM statement in line 1.

Figure 2 shows how programs are organized in the ZX/TS memory. The system variables point to the divisions between the successive areas of storage. When a block of lines is deleted from the program all of the memory locations past the deleted block up to E_LINE are moved down in memory to close up the gap. The system variables still point to their previous locations and they must be changed so that they point to the correct new addresses before any other operations are attempted. Otherwise, a crash will certainly be the unfortunate result. Once the process of relocating the BASIC code begins, it is not wise to try to access any BASIC variable, or to attemp to LIST or view the listing in any manner, until the system variables are changed to their new values. D_FILE, VARS, and E_LINE are critical in order to return the video display and variables area intact. The pointers END DELETE+1, START DELETE, DELTA, and POINTER are POKED into line one along with the system variables, so they can be manipulated by PEEKing and POKEing. Thus we can, in effect, have variables in storage without accessing the variable storage area. The system variables D_FILE (16396 and 16397), VARS (16400 and 16401), and E_LINE (16404 and 16405), are PEEKed, converted to decimal values, decreased by DELTA, and finally converted back to their hexadecimal equivalents and POKEd to the appropriate storage locations.

Up to this point, the entire program has been dedicated to preliminary manipulations and data input. Lines 122 to 160 should be entered now. Lines 124 thru 140 actually move all the code from the delete thru E_LINE. Lines 144 thru 154 restore the system pointers. Line 124 PEEKs the address POINTER and POKEs this byte to the address POINTER - DELTA. Line 126 checks to see if POINTER is less than E_LINE and either branches to line 130 to increment POINTER, or to line 144 which restores the system variables and ends the program.

The Block Delete program is now completed. Lines 920 thru 1100 can be added to check for proper operation or you can add several lines of your choice. Be sure to SAVE the program NOW, before attempting to run it unless you want to type it over. You will probably be disapointed to learn (as I was) that it takes over a minute to delete a block of lines using this routine. It is also over 2500 bytes...much too long for practical use. By deleting all of the REM statements except line 1, and all of the checkpoints, I was able to shorten the program to 1986 bytes. However, this did not decrease the execution time significantly and it still leaves one other problem. When you are through editing your program, you are left with 2500 bytes (and 81 lines) that you don't want.

The program can be re arranged so that it will delete itself except for a few lines) by moving the code from lines 124 to 158 to the beginning of the program and adding a couple of GOTO statements. However, the only method that I know to speed up the execution significantly is to resort to machine code. Therefore, in order to have a useful program, in terms of speed, length, and utility, I decided to convert the Block Move portion to machine code. This provides a very dramatic demonstration of the power and speed of machine code compared to BASIC. (In this case, the speed increases by more than 100 to 1, and 40 bytes replaces over 700 bytes of BASIC).

The program listed in Figure 3, is the machine code version of lines 124 thru 154. To enter it, and convert the BASIC program, first retype line 1 as:

```
1 REM (54 zeros and ENTER)
```

This provides 40 bytes for the machine code program and 14 bytes for storage locations. Next, delete all unecessary lines from the program. Delete lines 14, 16, 18, 20, 22, 24, 26, 28, 32, 40, 42, 58, 60, 62, 64, 66, 68, 70, 74, 82, 90, 98, 104, 110, 116, and 122. Then enter the following lines:

```
170 FOR N=16528 TO 16567
172 INPUT A$
175 POKE N, A$
185 NEXT N
190 STOP
```

SAVE the program at this point, then execute the code at 170 using GOTO 170 and input the codes listed under "Machine Code" in Figure 3, one at a time. You should receive the report code 9/190 after entering the code for RETURN, 201. Now enter the following lines:

```
 28 LET S=16509
124 GOTO 4
  2 GOTO 10
```

Figure 3. Block Move And Restore Pointers Routine

LABEL OR COMMENT	ASSEMBLY LISTING	ADDRESS	MACHINE CODE
Calculate # of Bytes	LD HL,(16404)	16528	42 20 64
	LD BC,(16524)	16531	237 75 140 64
	PUSH BC	16535	197
	AND A	16536	167
	SBC HL,BC	16537	237 66
	INC HL	16539	35
Store in BC	LD B,H	16540	68
	LD C,L	16541	77
Start Address To	POP HL	16542	225
Address Block Move	LD DE,(16526)	16543	237 91 142 64
	LDIR	16547	237 176
Restore D_FILE	LD HL,(16514)	16549	42 130 64
	LD (16396),HL	16552	34 12 64
Restore VARS	LD HL,(16516)	16555	42 132 64
	LD (16400),HL	16558	34 16 64
Restore E_LINE	LD HL,(16518)	16561	42 134 64
	LD (16404),HL	16564	34 20 64
Return	RET	16567	201

Total = 40 bytes

EDIT lines 4 to 12 in reverse order, changing the line numbers to:

```
12 to 26
10 to 24
 8 to 22
 6 to 20
 4 to 10
```

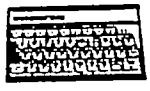
Delete lines, 4, 6, 8, and 12, and then enter the following lines:

```
4 RAND USR 16528
6 SLOW
8 STOP
```

SAVE the program and then carefully check it against Listing 2. The following lines can be added to check the machine code portion against that in Figure 3:

```
195 FOR N=16528 TO 16567
200 PRINT N;TAB 8;PEEK N
205 NEXT N
210 STOP
```

When the program checks out thru line 124, and the machine code checks with Figure 3, SAVE IT. Now test the program by deleting all lines past 124. This is the final version of the program SAVE it. You will find that it executes very, very fast, and that the length has been reduced to about 1300 bytes (41 lines), and you can delete the entire program past line 8 with no problems, leaving only 5 lines to delete manually when your new program is completed.



Listing 1.

```
  1 REM 00000000000000
  2 CLS
  4 PRINT TAB 5;"DELETE BLOCK=0
ENDS","ENTER LINE NUMBER: ST
ART";AT 2,19;"END"
  6 INPUT B
  8 PRINT AT 1,26;B
 10 INPUT E
 12 PRINT AT 2,28;E
 14 PRINT AT 5,5;"CORRECT=RETURN
"
 15 INPUT N$
 18 IF N$="N" THEN GOTO 2
 20 PRINT AT 7,0;"ENTER PROGRAM
START ADDRESS","ENTER LINE SPA
CE CHANGE","ENTER DELETE
START ADDRESS"
 22 INPUT S
 24 IF S=0 THEN LET S=16509
 26 PRINT AT 10,20;"START= ";S
 28 PAUSE 100
 30 FAST
 32 REM FIND BEGINNING
 34 IF 256*PEEK S+PEEK (S+1)>=B
 THEN GOTO 40
 36 LET S=S+4+PEEK (S+2)+256*PE
EK (S+3)
 38 GOTO 34
 40 PRINT AT 12,0;"LINE NUMBER
";B;" STARTS AT ";S
 42 REM FIND END
 44 LET T=S+4+PEEK (S+2)+256*PE
EK (S+3)
 46 LET TT=E
 48 IF 256*PEEK T+PEEK (T+1)>E
THEN GOTO 56
 50 LET TT=256*PEEK T+PEEK (T+1
)
 52 LET T=T+4+PEEK (T+2)+256*PE
EK (T+3)
 54 GOTO 48
 56 LET T=T-1
```

```
  58 PRINT AT 13,0;"LINE NUMBER
";TT;" ENDS AT ";T
  60 REM DELETE
  62 PRINT AT 15,0;"DELETE BLOCK
Y/N?"
  64 SLOW
  66 INPUT N$
  68 IF N$="N" THEN STOP
  70 FAST
  72 LET D=T+1-S
  74 REM D=NEW
  76 LET F=PEEK 16396+256*PEEK 1
6397-D
  78 POKE 16515,INT (F/256)
  80 POKE 16514,(F-256*PEEK 1651
5)
  82 REM VARS
  84 LET F=PEEK 16400+256*PEEK 1
6401-D
  86 POKE 16517,INT (F/256)
  88 POKE 16516,(F-256*PEEK 1651
7)
  90 REM E-LINE
  92 LET F=PEEK 16404+256*PEEK 1
6405-D
  94 POKE 16519,INT (F/256)
  96 POKE 16518,(F-256*PEEK 1651
9)
  98 REM END-PROG+1
 100 POKE 16525,INT ((T+1)/256)
 102 POKE 16524,T+1-256*PEEK 165
25
 104 REM STKBOT-PROG+1
 106 POKE 16527,INT (S/256)
 108 POKE 16526,S-256*PEEK 16527
 110 REM DEST
 112 POKE 16523,INT (D/256)
 114 POKE 16522,D-256*PEEK 16523
 116 REM POINTERS
 118 POKE 16520,PEEK 16524
 120 POKE 16521,PEEK 16525
 122 REM BC=NEWVAR
 124 POKE ((PEEK 16520+256*PEEK
16521)-(PEEK 16522+256*PEEK 1652
3)),PEEK (PEEK 16520+256*PEEK 15
521)
 126 IF (PEEK 16520+256*PEEK 165
21)<((PEEK 16518+256*PEEK 16519)
+(PEEK 16522+256*PEEK 16523)) TH
EN GOTO 130
 128 GOTO 144
 130 IF PEEK 16520<255 THEN GOTO
 132
 132 POKE 16520,0
 134 POKE 16521,PEEK 16521+1
 136 GOTO 124
 138 POKE 16520,PEEK 16520+1
 140 GOTO 124
 142 REM RESET SYSTEM POINTERS
 144 POKE 16396,PEEK 16514
 146 POKE 16397,PEEK 16515
 148 POKE 16400,PEEK 16516
 150 POKE 16401,PEEK 16517
 152 POKE 16404,PEEK 16518
 154 POKE 16405,PEEK 16519
 156 SLOW
 158 STOP
 160 REM INSERT PROGRAM HERE
 920 PRINT "LOWER LEFT CORNER"
 930 PRINT "X="
 940 INPUT X
 950 PRINT "Y="
 960 INPUT Y
 970 PRINT "W="
 980 INPUT W
 990 PRINT "H="
1000 INPUT H
1010 CLS
1015 FAST
1020 FOR J=X TO X+W
1030 PLOT J,Y
1040 PLOT J,Y+H
1050 NEXT J
1060 FOR J=Y TO Y+H
1070 PLOT X,J
1080 PLOT X+W,J
1090 NEXT J
1100 SLOW
```

Listing 2.

```
   1 REM 0000000000000E=RND GOS
UB 2?RNDVAL 3 GOSUB PI??? LPRINT
 GOSUB 5?RND GOSUB KEYRND6?RNDE?
RND6(RNDE?,RND6=RNDTAN
   2 GOTO 10
   4 RAND USR 16523
   6 SLOW
   8 STOP
  10 PRINT TAB 5;"DELETE BLOCK-D
ELINES","ENTER LINE NUMBER: ST
ART",AT 2,19;"END"
  20 INPUT B
  22 PRINT AT 1,28;B
  24 INPUT E
  26 PRINT AT 2,28;E
  28 LET S=16509
  30 FAST
  34 IF 256*PEEK S+PEEK (S+1)>=B
 THEN GOTO 40
  36 LET S=S+4+PEEK (S+2)+256*PE
EK (S+3)
  38 GOTO 34
  44 LET T=S+4+PEEK (S+2)+256*PE
EK (S+3)
  46 LET TT=E
  48 IF 256*PEEK T+PEEK (T+1)>E
THEN GOTO 56
  50 LET TT=256*PEEK T+PEEK (T+1
)
  52 LET T=T+4+PEEK (T+2)+256*PE
EK (T+3)
  54 GOTO 48
  56 LET T=T-1
  72 LET D=T+1-S
  76 LET F=PEEK 16396+256*PEEK 1
6397-D
  78 POKE 16515,INT (F/256)
  80 POKE 16514,(F-256*PEEK 1651
5)
  84 LET F=PEEK 16400+256*PEEK 1
6401-D
  86 POKE 16517,INT (F/256)
  88 POKE 16516,(F-256*PEEK 1651
7)
  92 LET F=PEEK 16404+256*PEEK 1
6405-D
  94 POKE 16519,INT (F/256)
  96 POKE 16518,(F-256*PEEK 1651
9)
 100 POKE 16525,INT ((T+1)/256)
 102 POKE 16524,T+1-256*PEEK 165
25
 106 POKE 16527,INT (S/256)
 108 POKE 16526,S-256*PEEK 16527
 112 POKE 16523,INT (D/256)
 114 POKE 16522,D-256*PEEK 16523
 118 POKE 16520,PEEK 16524
 120 POKE 16521,PEEK 16525
 124 GOTO 1
```

80 col. Printer Utility

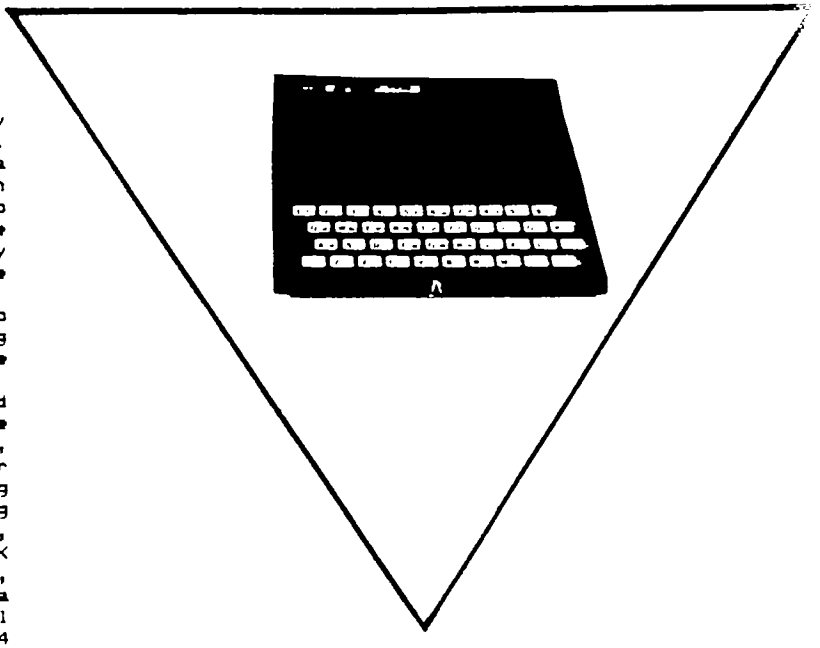
by Earl V. Dunnington

This is a short program for those with a ZX81/TS1000 (and TS1500), that use an 80 column printer, interface, and driver program, that will not do a screen dump. Note: see the REM statements for any instructions or comments.

```
   1 REM 80 COLUMN PRINTER,
TS1000/1500, SCREEN COPY,
TEXT ONLY.
PROGRAM BY EARL DUNNINGTON
   2 REM LINES 1 TO 55 AND 70
ARE FOR DEMONSTRATION ONLY
  10 FOR N=0 TO 21
  20 FOR M=32 TO 63
  30 PRINT CHR$ M;
  40 NEXT M
  50 NEXT N
  55 REM THE BASIC STATEMENT OF
LINE 60 MUST BE IN YOUR PROGRAM
  60 GOSUB 9880
  70 STOP
9880 DIM T$(32)
9890 LET N=1+(PEEK 16396+256*PEE
K 16397)
9900 LET S=N+726
9910 FOR M=1 TO 32
9920 LET T$(M)=CHR$ PEEK N
9930 LET N=N+1
9940 NEXT M
9950 LPRINT T$
9960 LET N=N+1
9970 IF N=S THEN RETURN
9980 GOTO 9910
```

"REAL COMPUTING POWER" FOR YOUR ZX81/TS1000

by Kenneth Tindle



The Sinclair ZX81 may just be the most cleverly designed computer ever made. I bought one by mail years ago, before Timex ever got involved. It's a great machine to learn computing on, as it can teach BASIC and machine language programming like no other. Some people, however, may overlook the true power of the Sinclair machines and automatically assume that they must get a fancier brand to have "real" computing power.

Don't you believe it! The ZX/TS computers do have four problem areas; these were engineering decisions made to keep the price low. Overcome these and you have a venerable Z80 based computer.

Two of these problems have already been covered by many books and magazines. That blasted little membrane keyboard is quite an annoyance (for a fix, see the first article in this book). The other problem is slow and unreliable cassette tape loading and saving of programs. This can be fixed by getting a data compression program like ZXLR-8 or Fastload, and putting it on EPROM, mapped into the 8K-16K block of memory. If you want to go the whole mile, Larken Electronics (RR#2 Navan, Ontario, Canada K4B-1H9) now has a DOS and controller card that will let you use one or two easily obtained 5 1/4 IBM-type disk drives. Imagine real honest-to goodness disk drives on a ZX81!

The other two problems only appear when connecting an external 16K RAM pack to a stock ZX81. These are thermal and RAM wobble crashes. The power supply is overloaded by fifteen extra chips it now must drive...there is just not enough heat sinking provided for that poor hard-working 7805 regulator. the lack of gold-plated fingers on the edge of the main PCB is like asking for momentary disconnection crashes. How to kill the problems and keep the RAM?

The Japanese have a solution. Their companies are responsible in large part for increasing capacities of static RAM chips while driving prices down. Hitachi has a chip numbered HM6264LP-15 which is an 8Kx8 static RAM, and is ideal for our purposes. Have your computer take two of these and in the morning you'll have 16K of internal RAM. As an extra plus, you do not have to stick with Sinclair's power supply scheme either. The Sinclair power packs require nine volts and five volts. This means you can remove the internal 7805, jumper its pins one and three (short Vin to Vout), and then hit the 9VDC jack with an external source of regulated five volts if your computer still overheats.

The best source I know of for these static RAMs is: JDR Microdevices, 1224 South Bascom Ave., San Jose, CA 95128. They also sell the 28-pin soldertail IC socket (which you will need).

When you get the new chips, handle them carefully to avoid zapping them with static charges on the pins. Keep them in their anti-static sleeves or foam, and when you install them, don't wear shoes or work around plastic sheets or wrappings. Only use a soldering iron with a grounded tip to prevent voltage transients from damaging the delicate CMOS inputs.

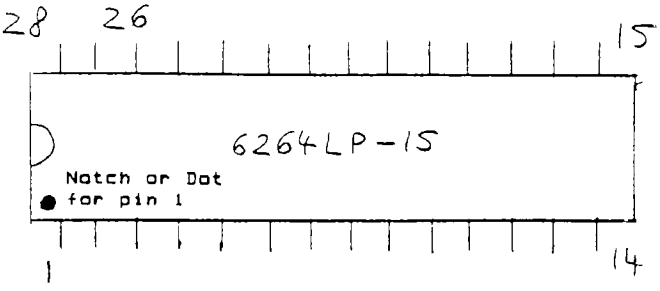
In addition to the iron, you are going to need a desoldering braid or wick, and some small diameter rosin core solder. Visit your local Radio Shack store for the transistor and two resistors shown on the schematic.

To start the conversion of either a ZX81 or a TS1000, all old RAM must be removed. These will be one or two chips in a ZX81. If there are two, you will see the number "2114" on their tops and they will be only half as wide as the others. If there is one, it will have "4118" on its top somewhere. A "4816" is 2K memory for a TS1000. In all cases, these chips will be located nearest the edge of the board opposite the side with the RF modulator for the TV. Any chip with designation MK36809 or anything with 40 pins must be left alone.

If you have only a desoldering braid, then use it to remove all solder from the foil side of the board. Then use it on top too, where the pins meet the upper board traces if there is no socket. While you are actually lifting the chips off the board, check to see that no traces are still soldered to the pins. If any traces are broken in the removal process, they will have to be jumpered later, but try to keep the board as intact as possible. This is sometimes not as easy as it sounds. Also remove the memory selection jumper L1 or L2 which will be close to the space for a 28-pin RAM.

After you have stripped off the old RAM, you must change the board to accept the pin-out of the new RAM. Pins 2,21,22,23 are the only changes. On the top, cut the trace to pin 2 on both sides. Also cut the trace between pins 20 and 22. On the bottom, cut the pad on pin 23 free without breaking the trace to pin 27. Install the 28 pin socket and tape it down from the component side. Solder all pins except 2,22 and 23. Remove the tape and install five jumpers. On the component side, put one jumper from trace 16B on the edge finger (use a plated through hole that is connected to that finger) to the center

Figure 1: IC Pin Layout



hole of L1-L2. Connect the L1 hole to the five volt side of R17. On the foil side, connect pin 2 with trace 14B. Connect pin 22 to trace 17A, and pin 23 with 15B. "A" refers to the fingers on the top, while "B" means the underside. I used wire wrap (small gauge solid) for all jumpers.

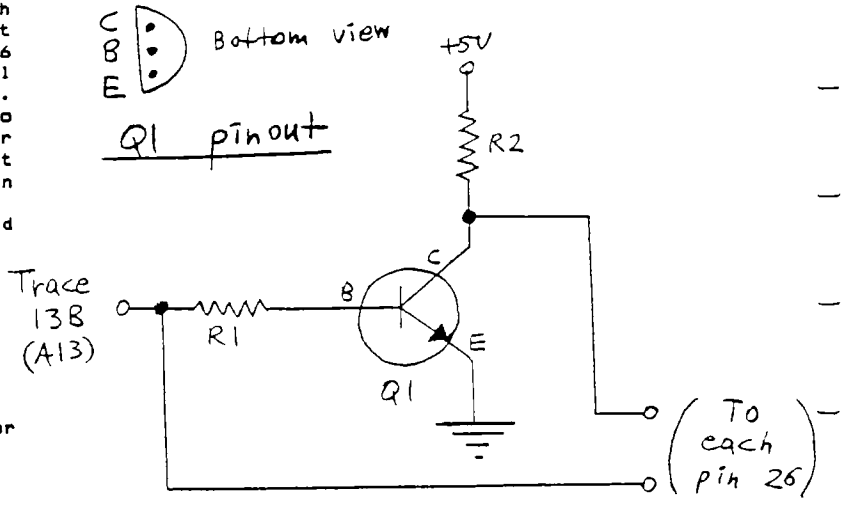
Now that you have the socket fully ready, install one RAM chip in the correct orientation with pin 26 turned up. Then install the second chip right on top of the first (piggy-back) with its pin 26 turned up as well. Now use your iron to solder all the pins together except the two you have bent up. Now connect the transistor inverter to these two pins according to Figure 2. Position the transistor and resistor leads so they can't short against anything. Note the transistor's pin configuration in Figure 2.

Now all that remains is to recase the board and enjoy crash free operation. Happy Computing!

Parts List

U1,U2 - Hitachi HM6264LP-15 8Kx8 static RAM
SO1 - 28 pin IC socket, solder tail
Q1 - 2N3904 NPN general purpose small signal transistor
R1 - 10K,5%,1/8 or 1/4 watt carbon film resistor
R2 - 2.2K,5%,1/8 or 1/4 watt carbon film resistor

Figure 2: Transistor Inverter



TIMEX COMPUTER CORPORATION

COMPUTER ENGINEERING

TECHNICAL MEMO # 0001

TITLE: TS1000 AND TS1500 I/O COMPATIBILITY

K. Paternostro
Nov. 7, 1983

I. I/O PORTS AND DISPLAY PROCESSING

The following I/O conditions are hardware defined in the SCLD of the two computers and are controlled by the firmware:

KBS/SLCR: This is a keyboard strobe signal which enables a reading of the keyboard and tape input. It also generates the vertical sync pulse necessary for TV displaying. The same I/O condition causes a latching of the line counter reset in hardware. This counter is used to keep track of the pixel line number within the characters being displayed. At the end of one frame the line counter requires a reset as the number of TV sweeps is not divisible by 8 (the number of pixel lines per character).

RLCR: This is a reset of the latch which enables the reset of the pixel line counter (0 to 7). In this way the line counter can only be reset by a setting of the reset latch (SLCR).

SNMIEN: This signal is an enable to the non-maskable interrupt generator in the SCLD. This signal is activated just prior to the creation of the top and bottom margins, then with each sync pulse that occurs an NMI is generated and counted. When the number of NMI's equals the desired number of lines for the margin, an RNMIEN is done and a jump into the display mode is made.

RNMIEN: This is a reset signal for the enable to a non-maskable interrupt generator in the SCLD. NMI's are used as a counter during the creation of the top and bottom margins in the TV display; therefore prior to the jump back into the display routine, the NMI generator must be disabled by this signal.

II. DECODING OF THE I/O PORTS IN THE TS1000

HARDWARE DECODE		FIRMWARE PORT USED
KBS/SLCR:	IORQ RD with A0=0	FE RD
RLCR:	ANY IORQ WR	FX WR; X=any value
SNMIEN:	IORQ WR with A0=0	FE WR
RNMIEN:	IORQ WR with A1=0	FD WR

III. DECODING OF THE I/O PORTS IN THE TS1500

HARDWARE DECODE		FIRMWARE PORT USED
KBS/SLCR:	IORQ RD with A0=0 and A7=1	FE RD
RLCR:	ANY IORQ WR with A7=1	FX WR; x=any value
SNMIEN:	IORQ WR with A0=0 and A7=1	FE WR
RNMIEN:	IORQ WR with A1=0 and A7=1	FD WR

IV. COMPATIBILITY RULES

Since A7 is used in the decoding of the TS1500 and is not used in the 1000 there are twice as many ports available in the former for use to third party vendors. Also, if any software written for the TS1000 attempts to access any of the above signals the condition of A7 must be met in order to achieve the same results in the 1500. For example in the TS1000, an instruction IN 00 will cause an enabling of the keyboard read, however it will not in the 1500 as A7=0. Therefore, the comparable instruction for the 1500 would be IN 80. For third party vendors who wish to port map external harware for both machines, the condition should be: A7 equal to zero, A0 and A1 equal to one for any I/O WR, and A7 equal to zero and A0 equal to one for any I/O RD's. Therefore, the following ports are available for I/O WR's:

03	07	0B	0F
13	17	1B	1F
23	27	2B	2F
33	37	3B	3F
43	47	4B	4F
53	57	5B	5F
63	67	6B	6F
73	77	7B	7F

NOTE: 73 and 77 are being used in the Modem.

The following are available for I/O RD's:

01	03	05	07	09	0B	0D	0F
11	13	15	17	19	1B	1D	1F
21	23	25	27	29	2B	2D	2F
31	33	35	37	39	3B	3D	3F
41	43	45	47	49	4B	4D	4F
51	53	55	57	59	5B	5D	5F
61	63	65	67	69	6B	6D	6F
71	73	75	77	79	7B	7D	7F

STRATEGIC FOOTBALL

by Michael Zerbo

STRATEGIC FOOTBALL is a two-player game for the ZX81/TS1000 with 16K RAM pack and the TS1500 computer. After typing in the following listing, make sure the program is in the SLOW mode before entering RUN. There are easy and hard levels of the program which can be chosen at the beginning of the game.

The game is timed and a two minute warning will be displayed near the game's end. Each player takes turns at the offense and makes the strategic decisions. The plays available are displayed on the upper right-hand side of the screen. Simply enter the number next to the play you want. The offense is shown as X's and the defense as O's. The computer controls the defense for both players.

After one of the players chooses his/her play, it will be acted out on the screen. To make the program simpler, and easier to create, I have both players face the same way on offense. The marker on top of the screen shows which player has the ball. When getting close to the goal line the players go into a goal line offense. You can choose the same plays as before, but they won't be acted out on the screen. Instead, the computer will simply say whether or not the offense was held. At the end of the game the computer will state which team won.

If you would rather not type in this listing, I will send you a copy on tape for $4.00. My address is: Michael Zerbo, 16 Colony Drive, West Sayville, NY 11796.

```
  1 REM "STRATEGIC FOOTBALL"
  2 REM BY MICHAEL ZERBO 3/4/84
  4 LET Y1=20
  5 LET R=6
  6 LET H=0
  7 LET O=0
  8 LET Y=20
  9 GOSUB 9900
 10 FOR F=3 TO 21
 11 PRINT AT F,0;"
                                                                
                                                                "
 12 NEXT F
 20 PRINT AT 21,1;"0 10 20 30 4
0 50 40 30 20 10 0"
 22 PRINT AT 20,1;"
                                                                "
 24 LET OU=1
 30 PRINT AT 3,1;"0 10 20 30 40
 50 40 30 20 10 0"
 31 LET ST=0
 32 PRINT TAB 1;"
                                                                "
 33 LET TEAM=1
 35 LET D=0
 36 POKE 16436,255
 37 POKE 16437,255
 40 PRINT AT 0,12;"STRATEGIC FO
OTBALL"
 41 SLOW
 50 PRINT AT 0,0;"TEAM 1";H;AT
 1,0;"TEAM 2";O
 54 PRINT AT (TEAM=1) 10,"  "
 55 PRINT AT TEAM-1,10;"X"
 56 GOSUB 9000
 60 PRINT AT 0,12;"OFFENSE- 1 R
UN    ";AT 1,12;"2 PASS 3 LONG B
OMB";AT 2,12;"4 SCREEN PASS 5 KI
CK"
 61 LET TIME=(65536-PEEK 16436-
256*PEEK 16437)/100
 64 IF R>=24 THEN GOSUB 9300
 65 INPUT O
 66 IF O<1 OR O>5 THEN GOTO 65
 70 GOSUB 7000
 72 IF TIME>=400 THEN GOTO 7999
 74 IF ST=0 AND TIME>=350 THEN
GOSUB 8400
 80 GOSUB 9000
 90 GOTO 60
9999 STOP
7000 IF O=1 THEN GOTO 7200
7002 PRINT AT 11,R;"  XO";AT 13,R
;"  XO";AT 12,R-1;"  X"
7015 PRINT AT 12,R;"  ";AT 12,R-2
;"X"
7030 PRINT AT 12,R-2;"  ";AT 10,R
-2;"  "
7031 PRINT AT 12,R-2;"X"
7032 PRINT AT 11,R-2;"X"
7034 PRINT AT 11,R-2;"  ";AT 11,R
-2;"X"
7040 LET K=INT (RND*4)+1
7042 GOTO 7100+(K*10)-10
7101 PRINT AT 11,R-2;"  X";AT 11,
R-1;"  ";AT 12,R-1;"X"
7102 PRINT AT 11,R;"  XO";AT 13,R
;"  XO";AT 12,R-1;"  X"
7103 LET Y=Y+7
7104 PRINT AT 12,R;"  X O"
7105 FOR F=0 TO 10
7106 PRINT AT 12,R+1;"  X";AT 12,
R+2;"O "
7107 NEXT F
7108 LET R=R+2
7109 RETURN
7111 LET Y=Y+3
7112 PRINT AT 11,R-2;"  X";AT 11,
R-1;"  ";AT 10,R-1;"X  O";AT 13,
R;"  XO"
7113 PRINT AT 18,R+3;"  "
7114 PRINT AT 19,R-1;"  X  O";A
T 10,R-1;"  XO"
7116 FOR F=0 TO 10
7117 PRINT AT 10,R;"  X";AT 10,R+
1;"O "
7118 NEXT F
7119 LET R=R+1
7120 RETURN
7122 PRINT AT 11,R-2;"  X"
7123 LET Y=Y+(INT (RND*3)-1)
7124 PRINT AT 11,R;"  O";AT 10,R
;"X"
7125 FOR F=0 TO 10
7126 PRINT AT 11,R-1;"  X";AT 11,
R;"O "
7127 NEXT F
7128 RETURN
7130 IF R<6 THEN GOTO 7040
7131 PRINT AT 11,R-2;"  X"
7132 PRINT AT 13,R-1;"XO  ";AT 1
1,R-1;"  "
7133 PRINT AT 10,R+1;"O "
7134 PRINT AT 13,R;"  "
7135 LET Y=Y-3
7136 FOR F=0 TO 10
7137 PRINT AT 12,R-1;"X";AT 12,R
+1;"O "
7138 NEXT F
7139 LET R=R-1
7140 RETURN
7200 IF O>2 THEN GOTO 7300
7210 PRINT AT 12,R-2;"  X"
7211 PRINT AT 12,R-3;"  ";AT 13,R
-2;"X";AT 13,R-1;"XO  ";AT 11,R-
1;"X"
7212 PRINT AT 13,R-3;"  ";AT 14,R
-2;"X";AT 14,R-2;"  ";AT 15,R-2;"
X"
7213 PRINT AT 19,R;"  X"
7214 PRINT AT 11,R-1;"  X";AT 8,R
-1;"  X";AT 19,R+2;"  X"
7215 PRINT AT 15,R-2;"  ";AT 19,R
+4;"  X";AT 15,R-6;"X"
7216 PRINT AT 15,R;"XO  ";AT 18,
R+6;"  ";AT 18,R+6;"X";AT 18,R+3;
"  ";AT 18,R+4;"O"
7217 PRINT AT 15,R-2;"  ";AT 15,R
-2;"X";AT 18,R+6;"  ";AT 17,R+4;"
X"
7218 PRINT AT 17,R+4;"  ";AT 16,R
+4;"X"
7219 LET X=INT (RND*24)+1
7220 IF X=24 THEN GOTO 7230
7221 IF E=1 AND X>9 THEN GOTO 72
50
7222 IF E=2 AND X>6 THEN GOTO 72
30
7224 LET Y=Y+10
7226 FOR F=R-1 TO R+3
7226 PRINT AT 16,F;"  "
7227 NEXT F
7228 PRINT AT 16,R+4;"X"
7229 PRINT AT 18,R+4;"  "
7230 FOR F=0 TO 10
7231 PRINT AT 16,R+3;"X ";AT 16,
R+3;"O "
7232 NEXT F
7233 LET R=R+3
7234 RETURN
7250 FOR F=R-1 TO R+5
7252 PRINT AT 16,F;"  "
7255 NEXT F
```

```
7256 RETURN
7230 PRINT AT 15,R+1;"█";AT 16,R
;"▣"
7231 PRINT AT 15,R-2;"X0█"
7232 IF R<6 THEN GOTO 7291
7234 LET Y=Y-7
7235 FOR F=0 TO 10
7236 PRINT AT 16,R-2;"▣█";AT 16,
R-2;"X█"
7238 NEXT F
7239 LET R=R-2
7290 RETURN
7291 PRINT AT 15,R-2;"X0▀";AT 15
,R-3;"X0█";AT 16,R-4;"X0▀"
7292 FOR F=0 TO 10
7293 PRINT AT 16,R-4;"▣█";AT 15,
R-4;"X"
7294 NEXT F
7295 PRINT AT 15,13;"SAFTEY"
7296 IF TEAM=1 THEN LET C=C+2
7297 IF TEAM=2 THEN LET H=H+2
7298 PRINT AT 15,0;"▒"
7299 GOTO 7509
7300 IF D>3 THEN GOTO 7400
7302 IF R>15 THEN GOTO 7210
7310 PRINT AT 12,R-2;"X██";AT 10
,R-2;"▀X ▣.▀"
7312 PRINT AT 13,R-1;"X0▀▀";AT 1
5,R-1;"X ▣.";AT 19,R-1;"▀ X█"
7314 PRINT AT 15,R;"▣.";AT 19,R+
1;"██X.";AT 18,R+3;"▀▣."
7316 PRINT AT 6,R-2;"▀▀X";AT 5,
R+1;"█.X";AT 7,R+1;"▀▣"
7318 PRINT AT 7,R+2;"▀█▣";AT 7,R
+4;"▀▣";AT 5,R+3;"██X";AT 5,R+
5;"▀X"
7320 PRINT AT 12,R-2;"█";AT 11,R
-2;"X";AT 11,R-2;"█";AT 10,R-2;"
X"
7322 PRINT AT 10,R-2;"█";AT 9,R-
2;"X";AT 9,R-2;"█"
7324 PRINT AT 8,R-2;"X";AT 9,R-1
;"X0██";AT 8,R-2;"█";AT 7,R-2;"X
";AT 7,R-2;"█";AT 5,R-3;"X"
7326 PRINT AT 5,R+8;"▀.X";AT 7,R
+8;"▀▣";AT 5,R+10;"▀X";AT 5,R+1
1;"▀X";AT 7,R+8;"▀▀▣"
7330 LET U=INT (RND*50)-1
7332 IF E=1 AND U>3 THEN GOTO 73
70
7333 IF E=2 AND U>3 THEN GOTO 73
70
7334 FOR F=R-1 TO R+12
7336 PRINT AT 6,F;"█."
7337 NEXT F
7340 PRINT AT 5,R+13;"X";AT 7,R+
10;"▀▣"
7342 FOR F=0 TO 10
7344 PRINT AT 7,R+12;"█";AT 6,R+
13;"▣";AT 5,R+13;"X"
7350 NEXT F
7351 LET Y=Y+43
7352 LET R=R+13
7354 RETURN
7370 FOR F=R-1 TO R+13
7372 PRINT AT 6,F;"█."
7374 NEXT F
7376 RETURN
7400 IF D<4 THEN GOTO 7500
7410 PRINT AT 12,R-2;"X.█"
7414 FOR F=11 TO 10 STEP -1
7416 PRINT AT F,R-2;"█"
7417 PRINT AT 11,R-2;"█";AT 10,R
-2;"X"
7420 NEXT F
7422 PRINT AT 11,R-1;"X0 ▀"
7423 IF E=1 AND (INT (RND*10)+1)
 =3 THEN GOTO 7450
7424 IF E=2 AND (INT (RND*10)+1)
=4 THEN GOTO 7450
7425 IF (INT (RND*3)+1)=1 THEN G
OTO 7444
7426 PRINT AT 10,R-2;"█X";AT 10,
R-1;"▀X";AT 10,R;"▀X"
7427 FOR F=0 TO 10
7428 PRINT AT 11,R;"█";AT 10,R+1
;"▣";AT 10,R+1;"X"
7429 NEXT F
7430 LET R=R+1
7432 LET Y=Y+3
7433 RETURN
7444 PRINT AT 10,R-2;"▀X";AT 10,
R-1;"▀X"
7445 FOR F=0 TO 10
7446 PRINT AT 11,R;"█";AT 10,R;"
▣";AT 10,R;"X"
7447 NEXT F
7448 RETURN
7450 PRINT AT 10,R-2;"▀X";AT 10,
R-1;"▀X";AT 10,R;"█X";AT 10,R+1;
"▀X"
7451 FOR F=0 TO 10
7452 PRINT AT 10,R+2;"▣█";AT 10,
R+2;"X"
7453 NEXT F
7454 LET R=R+2
7455 LET Y=Y+7
7460 RETURN
```

```
7500 PRINT AT 0,12;"(1) PUNT  OR
 ";AT 1,12;"(2) FIELD GOAL
 ";AT 2,12;"    ATTEMPT
"
7501 INPUT Y1
7502 IF Y1<1 OR Y1>2 THEN GOTO 7
501
7503 IF Y1=2 THEN GOTO 7522
7504 FOR F=5 TO 19
7505 PRINT AT F,1;"██████████████
█████████████"
7506 NEXT F
7507 LET PU=(Y>50)*(100-Y)+(Y<50
)*(80-Y)
7508 PRINT AT 6,2;PU;"-YARD PUNT
"
7509 FOR F=0 TO 40
7510 NEXT F
7511 LET D=0
7512 LET Y=20
7513 LET Y1=Y
7514 LET R=6
7515 LET TEAM=(TEAM=1)*2+(TEAM=2
)
7516 PRINT AT 0,12;"OFFENSE- 1 R
UN      ";AT 1,12;"2 PASS 3 LONG B
OMB";AT 2,12;"4 SCREEN PASS 5 KI
CK"
7517 GOTO 50
7522 PRINT AT 0,12;"    FIELD GOA
L      ";AT 1,12;"     ATTEMPT
";AT 2,12;"
"
7523 FOR F=5 TO 19
7524 PRINT AT F,1;"██████████████
█████████████"
7526 NEXT F
7528 PRINT AT 7,3;(100-Y)+17;"-Y
ARD ATTEMPT"
7529 FOR F=0 TO 20
7530 NEXT F
7531 PRINT AT 9,27;"▀█ ";AT 10,2
7;" █ ";AT 11,27;" ▀ ";AT 12,28;
" ";AT 13,28;" ";AT 14,28;" "
7532 PRINT AT 14,17;"X."
7533 LET J=INT (RND*10)+1
7535 LET V=0
7536 IF (100-Y)+17>50 THEN GOTO
7700
7550 IF (100-Y)+17<50 THEN GOTO
7570
7552 IF J<=3 THEN LET V=1
7553 GOTO 7700
7570 IF (100-Y)+17<40 THEN GOTO
7590
7572 IF J<=5 THEN LET V=1
7574 GOTO 7700
7590 IF J<=7 THEN LET V=1
7701 IF V=0 THEN GOTO 7750
7702 PRINT AT 14,18;"█";AT 13,19
;"█";AT 13,19;"█";AT 12,20;"█";A
T 12,20;"█";AT 11,22;"█";AT 11,2
2;"█";AT 10,25;"█";AT 10,25;"█";
AT 10,28;"█"
7705 PRINT AT 16,21;"GOOD"
7706 FOR F=0 TO 30
7707 NEXT F
7710 IF TEAM=1 THEN LET H=H+3
```

```
7711 IF TEAM=2 THEN LET C=C+3
7712 GOTO 7511
7750 PRINT AT 14,18;"██";AT 14,19;"█";AT 13,19;"█";AT 13,19;"█";AT 12,20;"█";AT 12,20;"█";AT 14,21;"█"
7751 LET E=2
7752 PRINT AT 16,20;"NO GOOD"
7753 FOR F=0 TO 20
7754 NEXT F
7755 LET D=0
7756 LET Y=100-Y
7757 LET R=INT (Y/3.333334)+1
7776 LET Y1=Y
7777 GOTO 7515
7999 FOR V=5 TO 19
8000 PRINT AT V,1;"██████████████████████████████"
8001 NEXT V
8003 PRINT AT 8,3;"GAME OVER-- TEAM ";(H>C)+(C>H)*2;" WINS"
8005 STOP
8400 FOR V=5 TO 19
8410 PRINT AT V,1;"██████████████████████████████"
8411 NEXT V
8420 PRINT AT 8,3;"TWO MINUTE WARNING"
8422 LET ST=1
8423 RETURN
9000 FOR V=5 TO 19
9002 PRINT AT V,1;"██████████████████████████████"
9003 NEXT V
9004 PRINT AT 12,R-1;"▒";AT 12,R;"▒"
9005 PRINT AT 10,R-2;"▒"
9006 PRINT AT 19,R;"▒";AT 5,R-1;"▒"
9007 PRINT AT 9,R;"▒";AT 11,R;"▒";AT 13,R;"▒";AT 15,R;"▒"
9008 PRINT AT 10,R+3;"▒";AT 7,R+3;"▒";AT 12,R+4;"▒";AT 18,R+3;"▒"
9010 PRINT AT 9,R+2;"▒";AT 11,R+2;"▒";AT 13,R+2;"▒";AT 15,R+2;"▒"
9011 IF Y>50 THEN GOTO 9050
9012 IF Y1+10<=Y THEN LET D=0
9013 IF Y1+10<=Y THEN LET Y1=Y
9018 LET D=D+1
9019 IF D>4 THEN GOTO 9200
9021 POKE 16418,0
9022 PRINT AT 22,0;D;" DOWN AND ";(Y1+10)-Y;" YARDS TO GO";AT 23,0;Y;" YARD LINE"
9023 FOR F=0 TO 40
9024 NEXT F
9025 POKE 16418,2
9030 RETURN
9050 LET N=50-(Y-50)
9052 IF Y1+10<=Y THEN LET D=0
9054 IF Y1+10<=Y THEN LET Y1=Y
9056 LET D=D+1
9057 IF D>4 THEN GOTO 9200
9058 POKE 16418,0
9060 PRINT AT 22,0;D;" DOWN AND ";(Y1+10)-Y;" YARDS TO GO";AT 23,0;N;" YARD LINE"
9062 FOR F=0 TO 40
9064 NEXT F
9066 POKE 16418,2
9070 RETURN
9200 LET TEAM=(TEAM=1)*2+(TEAM=2)
9201 LET Y=100-Y
9202 LET R=INT (Y/3.333334)+1
9204 LET D=0
9206 LET Y1=Y
9209 GOTO 50
9300 FOR F=5 TO 19
9302 PRINT AT F,1;"██████████████████████████████"
9303 NEXT F
9305 PRINT AT 5,3;"GOAL LINE OFFENSE"
9308 INPUT O
9309 IF O<1 OR O>5 THEN GOTO 9308
9310 IF O=5 THEN GOTO 7500
9311 LET U=INT (RND*10)+1
9312 IF U<=3 THEN GOTO 9350
9315 LET LN=(TEAM=1)*2+(TEAM=2)
9320 PRINT AT 8,3;"TEAM ";LN;" HOLDS OFFENSE"
9322 FOR F=0 TO 28
9324 NEXT F
9325 PRINT AT 8,3;"██████████████████████"
9330 GOTO 80
9350 IF TEAM=1 THEN LET H=H+7
9352 IF TEAM=2 THEN LET C=C+7
9400 FOR F=0 TO 10
9402 PRINT AT 12,3;"TEAM:";TEAM;" SCORES-TOUCHDOWN"
9404 PRINT AT 12,10;"SCORES TOUCHDOWN"
9405 NEXT F
9409 LET R=6
9410 LET TEAM=(TEAM=1)*2+(TEAM=2)
9411 LET Y=20
9412 LET D=0
9413 LET Y1=Y
9506 GOTO 50
9900 PRINT TAB 6;"STRATEGIC FOOTBALL"
9901 PRINT
9902 PRINT
9910 PRINT "THIS IS A TWO-PLAYER FOOTBALL    SIMULATION PROGRAM."
9911 PRINT ,,"THE STRATEGIC PLAYS OF EACH       TEAM ARE DECIDED BY YOU."
9915 PRINT AT 11,2;"-PRESS ANY KEY TO BEGIN-"
9920 IF INKEY$="" THEN GOTO 9920
9977 CLS
9980 PRINT "WHICH LEVEL DO YOU WANT?"
9981 PRINT ,,,"(1)EASY",,,,"(2)HARD"
9982 INPUT E
9983 IF E<1 OR E>2 THEN GOTO 9983
9984 CLS
9988 FAST
9989 RETURN
```

Is this the smart-ass computer programmer you were telling Mum about?

BUILD A SPECTRUM ROMSWITCH IN YOUR TS2068

by Gary Lessenberry

I recently purchased some Spectrum ROMs with the hope that I might be able to make my own ROMSWITCH-type circuit, without having to pay the high price of those that are commercially available. When I examined the TS2068's circuitry, I realized that this was an easier task than I had originally assumed! All that I needed was the Spectrum ROM, a toggle switch, two feet of insulated wire and two 10K ohm resistors.

To start the project, you first remove the top from your computer by removing screw on the bottom of the case. When you look inside, it will appear as in Figure A. You now remove the Timex ROM (U16). To remove it, gently pry it with a small screwdriver or knife insterted between the socket and the ROM.

With your Timex ROM removed, you place your Spectrum ROM directly over your Timex ROM with the notches in the same direction. There should only be a thin space between the two ROMs and all of their leads should be touching. Do not have a lot of space between these ROMs because clearance is critical when you reassemble your computer! You will now, very gently, bend pin 20 on both ROMs outward until they are perpendicular to the other pins. You may now solder all of the pins except pin 20. Be careful when soldering...allow about 30 seconds between the soldering of each pin so that you won't overheat and damage your ROMs. To pin 20, of each ROM, you will solder a piece of wire and one end of a 10K ohm resistor. The other end of each 10K ohm resistor will be soldered to pin 28 (+5VDC). The other end of the two wires that you have coming from pin 20 of the two ROMs will be soldered to the toggle switch. The toggle switch has three pins on it. Two of these pins are labled "ON". Solder one wire on each of these two pins. Another wire will be soldered on the middle pin of the toggle switch with the other end of that wire going to the circuit board and soldered to W1.

Figure A

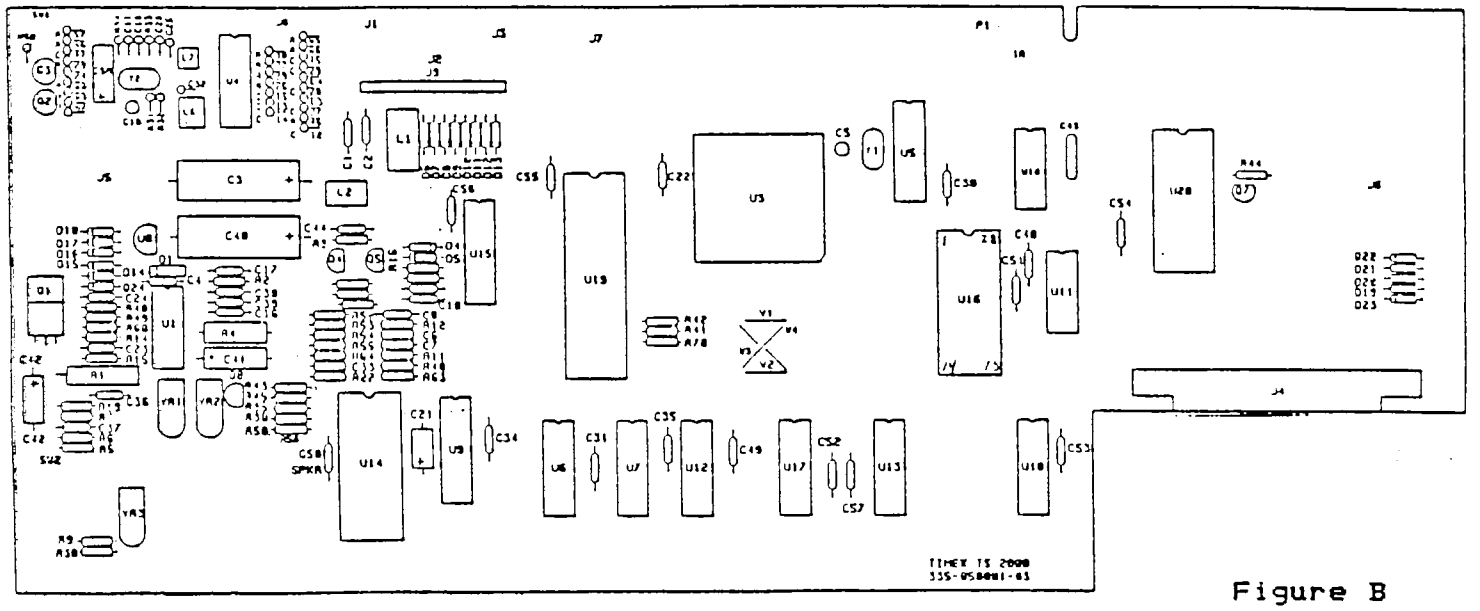
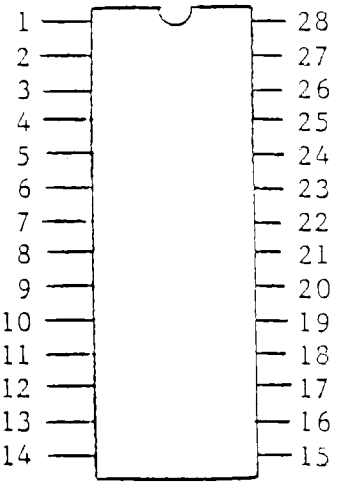
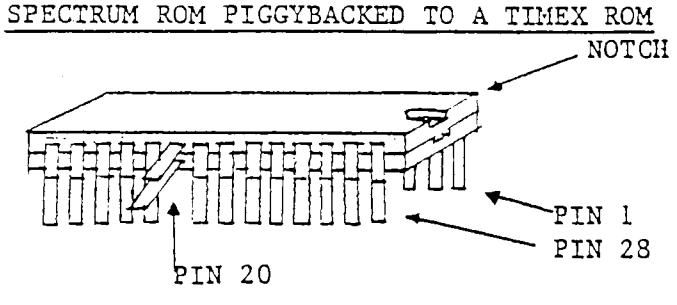


Figure B



SPECTRUM ROM PIGGYBACKED TO A TIMEX ROM

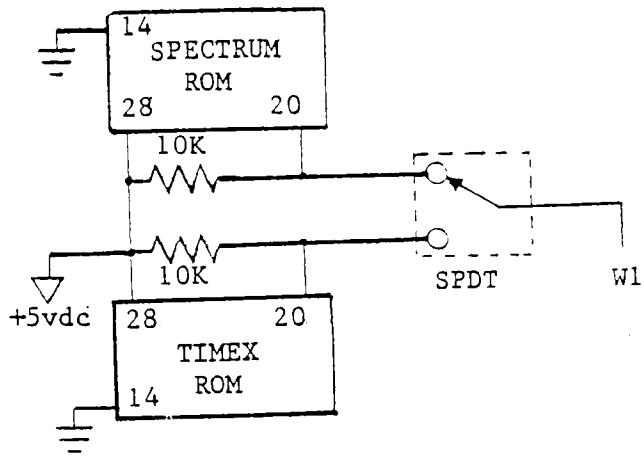


At this point, you may reinstall your ROM in it's socket. A hole must be drilled in the rear of your case for the mounting of the toggle switch. Once that your toggle switch is installed, you may replace the top of your computer. Be careful when reinstalling the top of the computer to ensure that you have proper clearance and that nothing is forced!

Once your computer is reassembled, you may test it out. You can tell which ROM is selected by the printout after intialization. When in Spectrum mode, you will get the Sinclair Copyright. When in 2068 mode, you will get both the Timex and Sinclair Copyrights.

Sources for Spectrum ROMs: 1. Zebra Systems, Inc., 78-06 Jamaica Ave., Woodhaven, NY 11421 (718) 296-2385; 2. G. Russell Electronics, RD1 Box 339, Centre Hall, PA 16828.

MODIFICATION SCHEMEMATIC



PARTS LIST

QTY	ITEM	SOURCE & CAT #
2	10K OHM RESISTORS	RADIO SHACK 271-1335
1	SPDT TOGGLE SWITCH	RADIO SHACK 275-6725
1	SPECTRUM ROM	ZEBRA SYSTEMS (no cat#
2ft	STRANDED WIRE, 25 GAUGE	

TS2068 AUTOMATIC EMERGENCY POWER SUPPLY

by Earl V. Dunnington

You are playing your favorite adventure game, using your word processor, working with a spreadsheet, entering files, or programming...and the 110 power voltage drops or momentarily cuts off. Hours of work lost!

Enter the Automatic Emergency Power Supply (see Figure No.1). The circuit works by voltage differentials. The external power supply for the TS2068 supplies DC current at a minimum 15 volts output with 110 volt AC input. Without a load, it actually tests out at 21 volts. Two 9 volt batteries wired in series provide 18 volts DC without a load. Two pairs of batteries connected in parallel provide sufficient amps to hold the program about 5 minutes depending on the type of battery and the number of accessories attached to the computer. The silicon

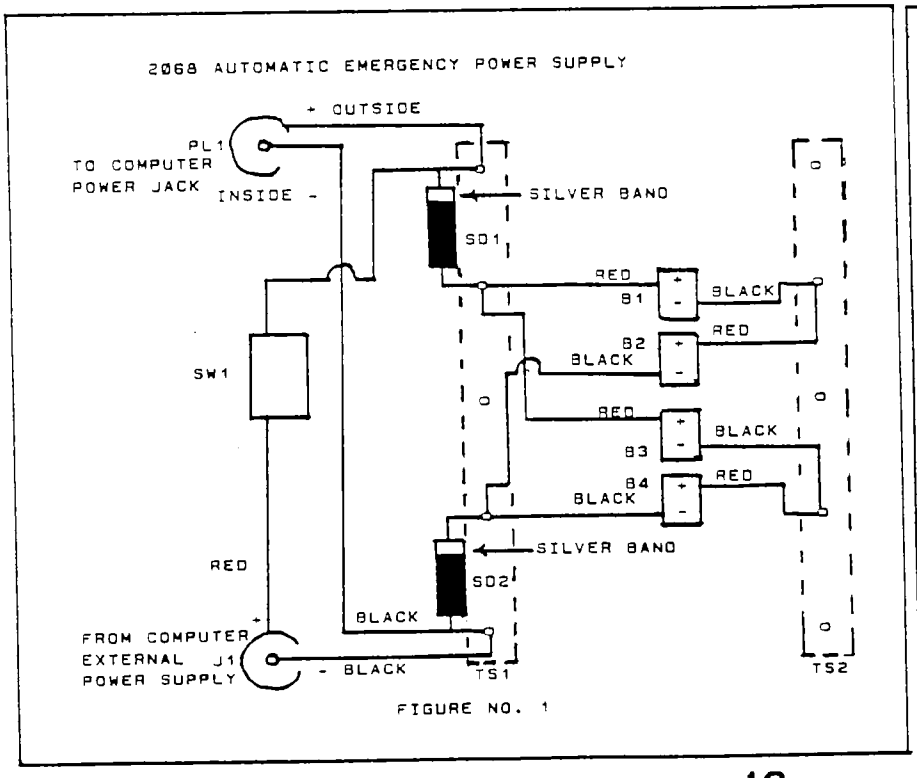


FIGURE NO. 1

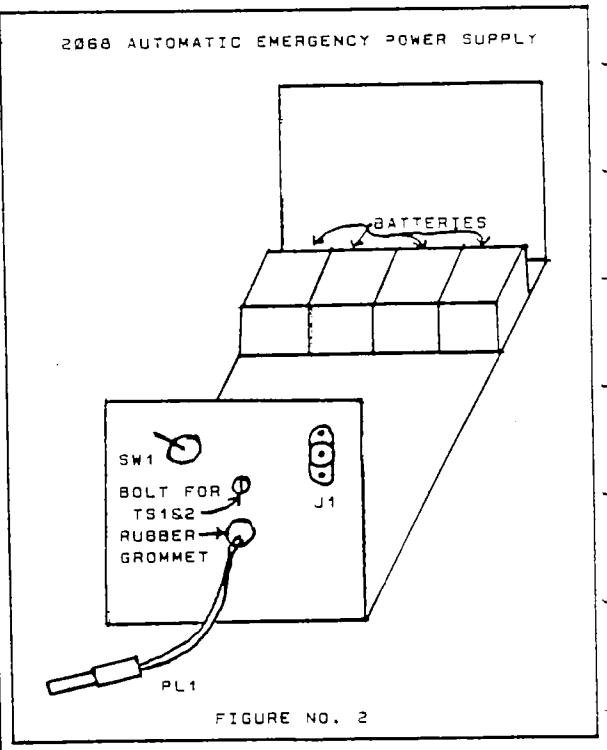


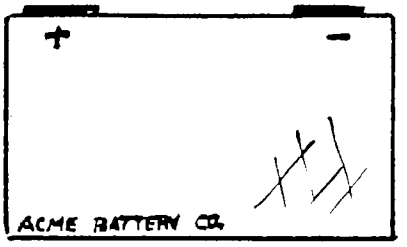
FIGURE NO. 2

diodes serve two purposes. They protect the batteries from being overcharged and blowing up by the current from the computer power supply. The silicon diodes also prevent current from flowing out of the batteries unless the voltage drops below the voltage of the batteries. The switch in the circuit is to prevent the batteries from leaking back into the external computer power supply when it is pulled from the 110 power. The power switch on the computer, when turned off, prevents the batteries from discharging by powering the computer when the external computer power supply is pulled.

When the project is completed and the batteries are installed, turn both switches off. Plug the external computer power supply into J1 and plug P1 into the power jack of the computer. Then plug the external computer power supply into the 110, turn SW1 on, and then the computer power switch. To power down, reverse the proceedure.

Parts List

B1,B2,B3,B4 9 volt rechargeable or non-rechargeable batteries
J1 jack (Radio Shack #274-1565)
P1 plug (Radio Shack #274-1567)
S1,S2 silicon diodes (Radio Shack #276-1114)
SW1 switch SPST (Radio Shack #275-624)
TS1,TS2 Terminal strips (Radio Shack #574-688)
9 volt battery snaps (Radio Shack #270-325), grommets, wire, project box



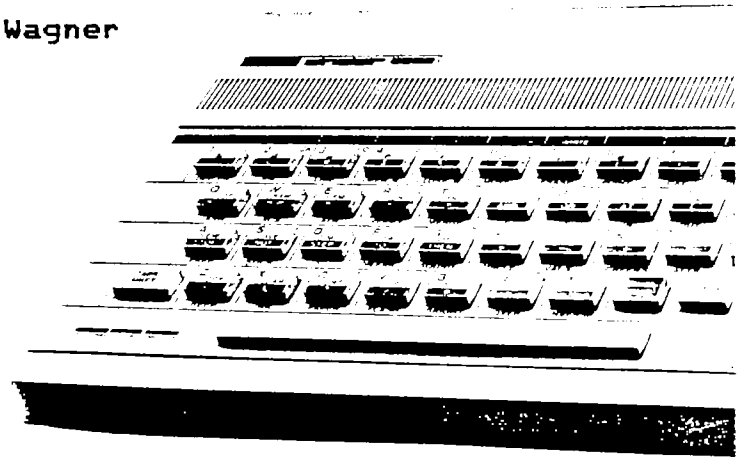
Additional Notes:

Silicon Diodes are easily damaged by too much heat. Heat sink the leads before soldering them.

Due to the dagers involved (Hydrogen Gas and Sulphuric Acid), I do not advocate the use of car storage batteries for emergency power. If you have a considerable number of accessories or wish to extend the program protection time, then use three six volt latern batteries wired in series instead of the two pairs of nine volt batteries. Be sure to locate them on an impervious plastic tray, so that any leakage will not damage your furniture or floor.

FIX THAT 2068 SPACE BAR

by Dick Wagner



Having experienced a hard-to-use space bar since I took delivery of my 2068 computer a long time ago (it seems), I recently decided that I had to stop all other activity and fix that space bar trouble. I'm inclined to believe that there are other 2068 computer users with a non-action right end space bar. If I hit the bar anywhere from the H key to the right end there just was no space action. If you have this problem you can test the space bar like this to determineif my fix will work on your computer- hold the left end of the space bar down not more than half the normal travel, quarter is better. Then press down on the right end to see if the space bar action occurs on the screen. If it works then my solution is worth trying.

Some time back I read that it was possible to get into the key pad switches by lifting the overlay for the keyboard. By carefully working one corner up with a sharp pointed knife it started to peal. It can be done without maring the case or overlay. As it lifted I used a clean flat tool like a screw driver to help pry. Carefully without stretching the overlay, I worked a pencil under the part I had lifted and across the keyboard so it wouldn't stick down again. Slowly it pealed up and was finally free. It was layed down sticky side up out of the way. This overlay is tough but by taking it easy it didn't stretch out of shape.

The next step was to remove the case top by removing the bottom screws. Then the keyboard ribbon cable was carefully removed from the bottom socket and the bottom half was put aside. Next the screws were removed from the underside of the top that held the keyboard box to the cover. The screws were kept segregated so that the proper ones would be used for each assembly process. There is a difference in screw lengths so be sure they go back in the correct holes.

Now I was at the key switch section. By removing the screws in the top of the box it was possible to get the thing apart. I soon discovered that this was best done with the keys down on a flat surface. There is a thin rubber membrane with a dimple and contact for each key. There are also some small holes in this membrane that fit over pins so the membrane can be properly positioned on assembly.

I set the key box upside down so the keys would not come out of their sockets, particularly the ENTER, and CAPS SHIFT keys that have the wire equalizers. More on this later. An inspection of the space bar construction showed it was easy to remove by first removing the 2 keys to the outside of the space bar arms. I found that long nose pliers could be used to grasp the arm near the hinge point and by twisting the pliers toward the opposite arm, the arm twisted enough so the hinge pin would be moved away from the hole and the arm lifted to be free of the bracket. The plastic arm is quite flexible so don't worry. In fact, this seems to be why the space bar doesn't work properly.

My first thought was to reinforce the bar itself by inserting a stiff piece of plastic into the space bar cavity and cementing it in place. This was slightly better but not worth the trouble. The solution was to cement a strip of thick paper or 0.025 thick aluminun in the spaces under the space bar arms. There is a cavity less than 1/8 inch wide that nicely held these strips. Don't use Krazy Glue because there may be some sliding to do before the glue sets. I found it was easier to put the glue on the strips than in the cavities.

The paper strips can be made from stock such as is used for Radio Shack blister packs. This stock measures very close to 0.025 inches, which is half way between 1/64 and 1/32 inch. I found that strips about 1/2 inch long could be handled nicely. Tweezers help in positioning the strips. The purpose of these spacers is to force the space bar closer to the contact points and compress the rubber buttons a little more. This effectively reduces the space bar travel. If spacers are too thick there may be continous contact or such little travel that the space bar may be accidently tripped.

After cementing the spacers in place I cleaned cement off the spacers as I didn't want the space bar stuck to anything. Next the space bar was put in place and the hinges reset by reverse tactics. The removed keys were placed in their proper sockets. Up-ending the keyboard so the keys could be checked seemed to be a good idea as I didn't want any reversed keys. I checked against a keyboard picture to be sure.

I mentioned the wire equalizers used on 3 keys. Should these become misplaced I found that the back of the wire must be under the loose holding clips and against the plastic housing. There is a socket made for this position.

The key switches were cleaned before assembly. A radio/TV contact cleaner and cotton QTips were handy for this. A dry swab on one end cleaned off the cleaner from all surfaces. DON'T spray!,just a little on a swab will do it. I also cleaned each carbon button with the dampened swab and then wiped. Don't get it on the rubber. One set of copper contacts had a dark mark on it so I used a Pink Pearl eraser to rub it and then cleaned again. I used a magnifying glass to check for foreign particles.

The parts were assembled in reverse order and with a couple of screws in place the keys were checked for proper movement and the 3 larger keys were moving properly. Raising the keybox up made it easier to get the rubber membrane in place over the pins. Holding the sandwitch together and turning it over, screws could be inserted.

The case assembly was made in reverse order. The difficult part was getting the wire ribbon back in the socket. Long nose pliers are required for this. I wrapped several turns of masking tape around each point so contact was at the tips. This helped. I did my best not to bend or damage the ribbon.

Before putting the case screws in place a test was run on all keys and in particular that the space bar was working right. Once operation was satisfactory, the case screws were installed and the overlay was carefully placed over the keyboard. I found that the cutouts are so close fitting that it was best to start at the center and lightly work toward the ends. With the ends just in the cavity I worked the overlay down, being sure that there were no hang-ups. Then by lighty pressing the membrane down everywhere, I was sure it was correct. Then I pressed harder all over for final contact.

For this effort I get a space everytime I press the space bar anywhere with any finger. No extra pressure is required either. Great!

Note: there is one step I haven't mentioned and I recommend doing it even tho I'm not sure it is helpful. I did a little filing of the underside of the right hand end of the space bar, from the hinge arm to the end as I thought it might be bottoming before contact was made. Pressure on the very end of the space bar will now just produce space action so maybe it helped. Watch how the space bar moves and you can see that the opposite end doesn't go down at all. This is partly due to the heavier rubber used for the spring action for the space bar switches.

LEAGUE SCHEDULER

by Jack L. DeLisle

After a friend told me he was responsible for scheduling the baseball games for his son's Little League, and it had taken him over 100 hours to complete the task, it looked like a job for my Timex Sinclair 2068.

What appears to be a random selection of team names in the array tables of the subroutines is actually a mathematical selection to insure an accurate mixing of teams. This mixing assures that no team ends up playing against themselves and at the same time allows for an approximately equal number of HOME and AWAY games.

How the program works:

LINE 40: Plays "Take me out to the ball game!"

LINES 50-80: Set TABS based on selection of printer to be used.

> TIMEX TS2040: TAB=16 for the listing 12 for length of team names. S$=40 dashes.
>
> FULL SIZED PRINTER: TAB=40 for listing and 32 for length of team names. S$=80 dashes.

LINES 90-130: Set DATE arrays for 16 dates, 15 characters long and allows input.

LINES 140-148: Allows entry of team name and centers for display.

LINES 150-180: Sets the number of teams, makes sure the number is even and between 4 and 14. Allows INPUT of team names. Names can be 12 characters long for TS2040 and 32 for full sized printer.

LINES 200-220: LPRINTS the heading (centered) and selects the subroutine based on the number of teams.

LINES 2000-7290: Match the teams for play and assign the "HOME" and "VISITOR" status.

LINES 7300-7350: Allows a new listing with the same dates, new dates, or exit from the program.

After typing the listing, GOTO 9000 to SAVE the program before running.

To view the schedules on the screen, you will need to change the LPRINT commands in the subroutines to PRINT commands. LEAGUE SCHEDULER will schedule up to 14 teams for up to 16 games. I you want a straight 16 game schedule, simply leave out the references to "WEEK 1", "WEEK 2", etc., as well as the references to "S$" in the subroutines of the program.

This program has saved countless hours in the scheduling of both baseball and football games, but could easily be adapted for almost any type of league play.

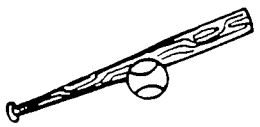
The program only gives a printed listing, however, TAB adjustments are made for either a full size printer or a TS2040. You will need to MERGE your printer driver software if a full size printer is used.

This program is available on cassette for $5.00 or AERCO disk for $6.00 (and also an IBM version on disk for $6.00)...for those who don't want to type in the listing. LEAGUE SCHEDULER can be adapted to schedule more teams and pairings, if so desired.

```
 10 REM LEAGUE SCHEDULER                Jack L. DeLisle 1986
 20 CLS : PRINT "THIS PROGRAM ALLOWS YOU TO      SCHEDULE UP TO 14 TEAMS FOR A
16GAME, 8 WEEK SCHEDULE.": PRINT : PRINT "IF THE NUMBER OF TEAMS IS NOT ANEVEN
NUMBER,SIMPLY INSERT A      'BYE' TO MAKE THE SCHEDULE COME OUT EVEN.": PRINT AT
 18,0;"PRESS ANY KEY TO CONTINUE": PAUSE 0:
 40 BEEP .5,0: BEEP .25,12: BEEP .25,9: BEEP .25,7: BEEP .25,4: BEEP 1,7: BEEP
.75,2: BEEP .5,0: BEEP .25,12: BEEP .25,9: BEEP .25,7: BEEP .25,4: BEEP 1,7
 50 CLS : PRINT "SELECT PRINTER FOR OUTPUT:": PRINT : PRINT : PRINT "T=TIMEX TS
-2040": PRINT : PRINT "F=FULL SIZE PRINTER": PAUSE 0
 60 IF INKEY$="T" OR INKEY$="t" THEN  LET Z=16: LET Y=12: LET S$="------------
------------------"
 70 IF INKEY$="F" OR INKEY$="f" THEN  LET Z=40: LET Y=32: LET S$="------------
--------------------------------------------------------------------"
 80 IF INKEY$<>"T" AND INKEY$<>"t" AND INKEY$<>"F" AND INKEY$<>"f" THEN  GO TO
50
 90 DIM D$(16,15)
100 CLS : PRINT "ENTER GAME DATES:": PRINT : PRINT :
110 FOR D=1 TO 16: INPUT D$(D): PRINT "DATE: ";D,D$(D): NEXT D
120 PRINT : PRINT "ALL CORRECT?": PAUSE 0
130 IF INKEY$="N" OR INKEY$="n" THEN  INPUT "What date is wrong?";X: CLS : PRIN
T "ENTER NEW DATE ";X: INPUT D$(X): CLS : FOR D=1 TO 16: PRINT "DATE: ";D,D$(D)
: NEXT D: GO TO 120
140 CLS : INPUT "ENTER LEAGUE NAME:                ";L$
145 PRINT AT 11,(16-(LEN L$)/2);L$''TAB 12;"CORRECT?": PAUSE 0
148 IF INKEY$="N" OR INKEY$="n" THEN  GO TO 140
150 CLS : PRINT "How many teams are to be listed?": PRINT : PRINT : INPUT T
152 IF INT (T/2)<>INT T/2 THEN  CLS : PRINT AT 11,6: FLASH 1;"NUMBER MUST BE EV
EN": BEEP .5,-20: PAUSE 120: GO TO 150
153 IF T<4 THEN  CLS : PRINT AT 11,6: FLASH 1;"NUMBER TOO SMALL": BEEP .5,-20:
PAUSE 120: GO TO 150
154 IF T>14 THEN  CLS : PRINT AT 11,6: FLASH 1;"NUMBER TOO LARGE": BEEP .5,-20:
 PAUSE 120: GO TO 150
155 PRINT T: DIM T$(T,Y): PAUSE 60
160 CLS : PRINT "ENTER TEAM NAMES"'': FOR X=1 TO T: INPUT T$(X): PRINT "TEAM ";
X,T$(X): NEXT X
170 PRINT : PRINT "ALL CORRECT?": PAUSE 0
180 IF INKEY$="N" OR INKEY$="n" THEN  INPUT "What team is wrong? ";N: INPUT "EN
TER CORRECT NAME:              ";T$(N): CLS : FOR X=1 TO T: PRINT "TEAM ";X,T$(X
): NEXT X: GO TO 170
190 CLS : LPRINT TAB Z-LEN L$/2;L$''
200 FOR X=1 TO Z*2: LPRINT "=";: NEXT X
210 LPRINT TAB (Y/4);"HOME TEAM";TAB (Y+5);"VISITORS"
220 GO TO t*500
```

```
2000 LPRINT '"WEEK 1:"'D$(1)'T$(1);TAB Z;T$(2)'T$(3);TAB Z;T$(4)''D$(2)'T$(3);TA
B Z;T$(1)'T$(4);TAB Z;T$(2)
2010 LPRINT
2020 LPRINT S$
2030 LPRINT '"WEEK 2:"'D$(3)'T$(2);TAB Z;T$(3)'T$(1);TAB Z;T$(4)''D$(4)'T$(3);TA
B Z;T$(4)'T$(2);TAB Z;T$(1)'
2040 LPRINT
2050 LPRINT S$
2060 LPRINT '"WEEK 3:"'D$(5)'T$(4);TAB Z;T$(2)'T$(1);TAB Z;T$(3)''D$(6)'T$(1);TA
B Z;T$(4)'T$(3);TAB Z;T$(2)'
2070 LPRINT
2080 LPRINT S$
2090 LPRINT '"WEEK 4:"'D$(7)'T$(2);TAB Z;T$(1)'T$(4);TAB Z;T$(3)''D$(8)'T$(1);TA
B Z;T$(3)'T$(2);TAB Z;T$(4)
2100 LPRINT
2110 LPRINT S$
2120 LPRINT '"WEEK 5:"'D$(9)'T$(3);TAB Z;T$(2)'T$(4);TAB Z;T$(1)''D$(10)'T$(4);T
AB Z;T$(3)'T$(1);TAB Z;T$(2)
2130 LPRINT
2140 LPRINT S$
2150 LPRINT '"WEEK 6:"'D$(11)'T$(2);TAB Z;T$(4)'T$(3);TAB Z;T$(1)''D$(12)'T$(4);
TAB Z;T$(1)'T$(2);TAB Z;T$(3)
2160 LPRINT
2170 LPRINT S$
```

```
2180 LPRINT '"WEEK 7:"'D$(13)'T$(1);TAB Z;T$(2)'T$(3);TAB Z;T$(4)''D$(14)'T$(3);
TAB Z;T$(1)'T$(4);TAB Z;T$(2)
2190 LPRINT
2200 LPRINT S$
2210 LPRINT '"WEEK 8:"'D$(15)'T$(4);TAB Z;T$(2)'T$(1);TAB Z;T$(3)''D$(16)'T$(1);
TAB Z;T$(4)'T$(3);TAB Z;T$(2)'
2220 GO TO 7300
3000 LPRINT '"WEEK 1:"'D$(1)'T$(1);TAB Z;T$(2)'T$(3);TAB Z;T$(4)'T$(5);TAB Z;T$(
6)''D$(2)'T$(5);TAB Z;T$(4)'T$(1);TAB Z;T$(6)'T$(2);TAB Z;T$(3)'
3010 LPRINT
3020 LPRINT S$
3030 LPRINT '"WEEK 2:"'D$(3)'T$(3);TAB Z;T$(1)'T$(2);TAB Z;T$(5)'T$(6);TAB Z;T$(
4)''D$(4)'T$(2);TAB Z;T$(6)'T$(4);TAB Z;T$(1)'T$(3);TAB Z;T$(5)'
3040 LPRINT
3050 LPRINT S$
3060 LPRINT '"WEEK 3:"'D$(5)'T$(1);TAB Z;T$(5)'T$(6);TAB Z;T$(3)'T$(4);TAB Z;T$(
2)''D$(6)'T$(3);TAB Z;T$(2)'T$(4);TAB Z;T$(5)'T$(1);TAB Z;T$(6)'
3070 LPRINT
3080 LPRINT S$
3090 LPRINT '"WEEK 4:"'D$(7)'T$(5);TAB Z;T$(6)'T$(1);TAB Z;T$(2)'T$(3);TAB Z;T$(
4)''D$(8)'T$(1);TAB Z;T$(4)'T$(5);TAB Z;T$(3)'T$(6);TAB Z;T$(2)'
3100 LPRINT
3110 LPRINT S$
3120 LPRINT '"WEEK 5:"'D$(9)'T$(6);TAB Z;T$(3)'T$(2);TAB Z;T$(4)'T$(5);TAB Z;T$(
1)''D$(10)'T$(2);TAB Z;T$(5)'T$(3);TAB Z;T$(1)'T$(4);TAB Z;T$(6)'
3130 LPRINT
3140 LPRINT S$
3150 LPRINT '"WEEK 6:"'D$(11)'T$(4);TAB Z;T$(3)'T$(6);TAB Z;T$(5)'T$(2);TAB Z;T$
(1)''D$(12)'T$(6);TAB Z;T$(1)'T$(3);TAB Z;T$(2)'T$(4);TAB Z;T$(5)'
3160 LPRINT
3170 LPRINT S$
3180 LPRINT '"WEEK 7:"'D$(13)'T$(5);TAB Z;T$(2)'T$(4);TAB Z;T$(6)'T$(1);TAB Z;T$
(3)''D$(14)'T$(1);TAB Z;T$(4)'T$(5);TAB Z;T$(3)'T$(6);TAB Z;T$(2)'
3190 LPRINT
3200 LPRINT S$
3210 LPRINT '"WEEK 8:"'D$(15)'T$(3);TAB Z;T$(6)'T$(2);TAB Z;T$(4)'T$(5);TAB Z;T$
(1)''D$(16)'T$(5);TAB Z;T$(4)'T$(6);TAB Z;T$(1)'T$(2);TAB Z;T$(3)'
3220 GO TO 7300
4000 LPRINT '"WEEK 1:"'D$(1)'T$(1);TAB Z;T$(2)'T$(3);TAB Z;T$(4)'T$(5);TAB Z;T$(
6)'T$(7);TAB Z;T$(8)''D$(2)'T$(6);TAB Z;T$(8)'T$(5);TAB Z;T$(7)'T$(2);TAB Z;T$(
4)'T$(1);TAB Z;T$(3)'
4010 LPRINT
4020 LPRINT S$
4030 LPRINT '"WEEK 2:"'D$(3)'T$(5);TAB Z;T$(4)'T$(1);TAB Z;T$(8)'T$(7);TAB Z;T$(
3)'T$(2);TAB Z;T$(6)''D$(4)'T$(3);TAB Z;T$(6)'T$(7);TAB Z;T$(2)'T$(1);TAB Z;T$(
5)'T$(8);TAB Z;T$(4)'
4040 LPRINT
4050 LPRINT S$
4060 LPRINT '"WEEK 3:"'D$(5)'T$(7);TAB Z;T$(1)'T$(4);TAB Z;T$(6)'T$(3);TAB Z;T$(
8)'T$(5);TAB Z;T$(2)''D$(6)'T$(2);TAB Z;T$(3)'T$(8);TAB Z;T$(5)'T$(4);TAB Z;T$(
1)'T$(6);TAB Z;T$(7)'
4070 LPRINT
4080 LPRINT S$
4090 LPRINT '"WEEK 4:"'D$(7)'T$(4);TAB Z;T$(7)'T$(6);TAB Z;T$(1)'T$(8);TAB Z;T$(
2)'T$(3);TAB Z;T$(5)''D$(8)'T$(5);TAB Z;T$(6)'T$(7);TAB Z;T$(8)'T$(3);TAB Z;T$(
4)'T$(1);TAB Z;T$(2)'
4100 LPRINT
4110 LPRINT S$
4120 LPRINT '"WEEK 5:"'D$(9)'T$(2);TAB Z;T$(4)'T$(1);TAB Z;T$(3)'T$(5);TAB Z;T$(
7)'T$(6);TAB Z;T$(8)''D$(10)'T$(7);TAB Z;T$(3)'T$(2);TAB Z;T$(6)'T$(1);TAB Z;T$
(8)'T$(5);TAB Z;T$(4)'
4130 LPRINT
4140 LPRINT S$
4150 LPRINT '"WEEK 6:"'D$(11)'T$(1);TAB Z;T$(5)'T$(8);TAB Z;T$(4)'T$(7);TAB Z;T$
(2)'T$(3);TAB Z;T$(6)''D$(12)'T$(3);TAB Z;T$(8)'T$(5);TAB Z;T$(2)'T$(4);TAB Z;T
$(6)'T$(7);TAB Z;T$(1)'
4160 LPRINT
4170 LPRINT S$
4180 LPRINT '"WEEK 7:"'D$(13)'T$(4);TAB Z;T$(1)'T$(6);TAB Z;T$(7)'T$(8);TAB Z;T$
(5)'T$(2);TAB Z;T$(3)''D$(14)'T$(8);TAB Z;T$(2)'T$(3);TAB Z;T$(5)'T$(6);TAB Z;T
$(1)'T$(4);TAB Z;T$(7)'
4190 LPRINT
4200 LPRINT S$
4210 LPRINT '"WEEK 8:"'D$(15)'T$(4);TAB Z;T$(3)'T$(2);TAB Z;T$(1)'T$(8);TAB Z;T$
(7)'T$(6);TAB Z;T$(5)''D$(16)'T$(7);TAB Z;T$(5)'T$(8);TAB Z;T$(6)'T$(3);TAB Z;T
$(1)'T$(4);TAB Z;T$(2)'
4220 GO TO 7300
5000 LPRINT '"WEEK 1:"'D$(1)'T$(1);TAB Z;T$(2)'T$(3);TAB Z;T$(4)'T$(5);TAB Z;T$(
6)'T$(7);TAB Z;T$(8)'T$(9);TAB Z;T$(10)'
5010 LPRINT 'D$(2)'T$(7);TAB Z;T$(3)'T$(1);TAB Z;T$(6)'T$(2);TAB Z;T$(9)'T$(5);T
AB Z;T$(10)'T$(8);TAB Z;T$(4)'
5020 LPRINT
5030 LPRINT S$
5040 LPRINT '"WEEK 2:"'D$(3)'T$(4);TAB Z;T$(5)'T$(9);TAB Z;T$(8)'T$(10);TAB Z;T$
(1)'T$(3);TAB Z;T$(2)'T$(6);TAB Z;T$(7)'
5050 LPRINT 'D$(4)'T$(9);TAB Z;T$(1)'T$(5);TAB Z;T$(3)'T$(4);TAB Z;T$(7)'T$(8);T
AB Z;T$(6)'T$(10);TAB Z;T$(2)'
5060 LPRINT
5070 LPRINT S$
```



```
5080 LPRINT '"WEEK 3:"'D$(5)'T$(10);TAB Z;T$(7)'T$(6);TAB Z;T$(2)'T$(8);TAB Z;T$
(3)'T$(4);TAB Z;T$(1)'T$(5);TAB Z;T$(9)'
5090 LPRINT 'D$(6)'T$(5);TAB Z;T$(8)'T$(4);TAB Z;T$(10)'T$(7);TAB Z;T$(2)'T$(6);
TAB Z;T$(9)'T$(1);TAB Z;T$(3)'
5100 LPRINT
5110 LPRINT S$
5120 LPRINT '"WEEK 4:"'D$(7)'T$(6);TAB Z;T$(4)'T$(7);TAB Z;T$(9)'T$(1);TAB Z;T$(
5)'T$(10);TAB Z;T$(3)'T$(2);TAB Z;T$(8)'
5130 LPRINT 'D$(8)'T$(3);TAB Z;T$(9)'T$(8);TAB Z;T$(1)'T$(6);TAB Z;T$(10)'T$(2);
TAB Z;T$(4)'T$(7);TAB Z;T$(5)'
5140 LPRINT
5150 LPRINT S$
5160 LPRINT '"WEEK 5:"'D$(9)'T$(8);TAB Z;T$(10)'T$(2);TAB Z;T$(5)'T$(9);TAB Z;T$
(4)'T$(1);TAB Z;T$(7)'T$(3);TAB Z;T$(6)'
5170 LPRINT 'D$(10)'T$(4);TAB Z;T$(3)'T$(10);TAB Z;T$(9)'T$(2);TAB Z;T$(1)'T$(6)
;TAB Z;T$(5)'T$(8);TAB Z;T$(7)'
5180 LPRINT
5190 LPRINT S$
5200 LPRINT '"WEEK 6:"'D$(11)'T$(6);TAB Z;T$(1)'T$(4);TAB Z;T$(8)'T$(3);TAB Z;T$
(7)'T$(9);TAB Z;T$(2)'T$(10);TAB Z;T$(5)'
5210 LPRINT 'D$(12)'T$(8);TAB Z;T$(9)'T$(7);TAB Z;T$(6)'T$(5);TAB Z;T$(4)'T$(1);
TAB Z;T$(10)'T$(2);TAB Z;T$(3)'
5220 LPRINT
5230 LPRINT S$
5240 LPRINT '"WEEK 7:"'D$(13)'T$(3);TAB Z;T$(5)'T$(2);TAB Z;T$(10)'T$(1);TAB Z;T
$(9)'T$(7);TAB Z;T$(4)'T$(6);TAB Z;T$(8)'
5250 LPRINT 'D$(14)'T$(2);TAB Z;T$(6)'T$(9);TAB Z;T$(5)'T$(7);TAB Z;T$(10)'T$(3)
;TAB Z;T$(8)'T$(1);TAB Z;T$(4)'
5260 LPRINT
5270 LPRINT S$
5280 LPRINT '"WEEK 8:"'D$(15)'T$(10);TAB Z;T$(4)'T$(3);TAB Z;T$(1)'T$(8);TAB Z;T
$(5)'T$(2);TAB Z;T$(7)'T$(9);TAB Z;T$(6)'
5290 LPRINT 'D$(16)'T$(9);TAB Z;T$(7)'T$(8);TAB Z;T$(2)'T$(4);TAB Z;T$(6)'T$(5);
TAB Z;T$(1)'T$(3);TAB Z;T$(10)'
5300 GO TO 7300
6000 LPRINT '"WEEK 1:"'D$(1)'T$(1);TAB Z;T$(2)'T$(3);TAB Z;T$(4)'T$(5);TAB Z;T$(
6)'T$(7);TAB Z;T$(8)'T$(9);TAB Z;T$(10)'T$(11);TAB Z;T$(12)'
6010 LPRINT 'D$(2)'T$(4);TAB Z;T$(5)'T$(6);TAB Z;T$(2)'T$(12);TAB Z;T$(3)'T$(9);
TAB Z;T$(11)'T$(1);TAB Z;T$(7)'T$(10);TAB Z;T$(8)'
6020 LPRINT
6030 LPRINT S$
6040 LPRINT '"WEEK 2:"'D$(3)'T$(9);TAB Z;T$(3)'T$(1);TAB Z;T$(10)'T$(11);TAB Z;T
$(4)'T$(5);TAB Z;T$(12)'T$(8);TAB Z;T$(2)'T$(6);TAB Z;T$(7)'
6050 LPRINT 'D$(4)'T$(7);TAB Z;T$(12)'T$(5);TAB Z;T$(8)'T$(9);TAB Z;T$(2)'T$(10)
;TAB Z;T$(4)'T$(11);TAB Z;T$(6)'T$(1);TAB Z;T$(3)'
6060 LPRINT
6070 LPRINT S$
6080 LPRINT '"WEEK 3:"'D$(5)'T$(11);TAB Z;T$(8)'T$(9);TAB Z;T$(7)'T$(1);TAB Z;T$
(5)'T$(6);TAB Z;T$(3)'T$(10);TAB Z;T$(12)'T$(2);TAB Z;T$(4)'
6090 LPRINT 'D$(6)'T$(10);TAB Z;T$(6)'T$(11);TAB Z;T$(1)'T$(3);TAB Z;T$(8)'T$(12
);TAB Z;T$(2)'T$(7);TAB Z;T$(4)'T$(9);TAB Z;T$(5)'
6100 LPRINT
6110 LPRINT S$
6120 LPRINT '"WEEK 4:"'D$(7)'T$(5);TAB Z;T$(7)'T$(4);TAB Z;T$(12)'T$(10);TAB Z;T
$(2)'T$(1);TAB Z;T$(9)'T$(6);TAB Z;T$(8)'T$(3);TAB Z;T$(11)'
6130 LPRINT 'D$(8)'T$(9);TAB Z;T$(12)'T$(10);TAB Z;T$(5)'T$(11);TAB Z;T$(7)'T$(4
);TAB Z;T$(6)'T$(2);TAB Z;T$(3)'T$(8);TAB Z;T$(1)'
6140 LPRINT
6150 LPRINT S$
6160 LPRINT '"WEEK 5:"'D$(9)'T$(6);TAB Z;T$(1)'T$(2);TAB Z;T$(11)'T$(8);TAB Z;T$
(12)'T$(3);TAB Z;T$(5)'T$(4);TAB Z;T$(9)'T$(7);TAB Z;T$(10)'
6170 LPRINT 'D$(10)'T$(3);TAB Z;T$(10)'T$(9);TAB Z;T$(8)'T$(1);TAB Z;T$(4)'T$(2)
;TAB Z;T$(7)'T$(5);TAB Z;T$(11)'T$(12);TAB Z;T$(6)'
6180 LPRINT
6190 LPRINT S$
6200 LPRINT '"WEEK 6:"'D$(11)'T$(8);TAB Z;T$(4)'T$(7);TAB Z;T$(3)'T$(6);TAB Z;T$
(9)'T$(11);TAB Z;T$(10)'T$(12);TAB Z;T$(1)'T$(5);TAB Z;T$(2)'
6210 LPRINT 'D$(12)'T$(6);TAB Z;T$(5)'T$(2);TAB Z;T$(1)'T$(12);TAB Z;T$(11)'T$(4
);TAB Z;T$(3)'T$(8);TAB Z;T$(7)'T$(10);TAB Z;T$(9)'
6220 LPRINT
6230 LPRINT S$
6240 LPRINT '"WEEK 7:"'D$(13)'T$(3);TAB Z;T$(12)'T$(5);TAB Z;T$(4)'T$(8);TAB Z;T
$(10)'T$(2);TAB Z;T$(6)'T$(11);TAB Z;T$(9)'T$(7);TAB Z;T$(1)'
6250 LPRINT 'D$(14)'T$(4);TAB Z;T$(11)'T$(3);TAB Z;T$(9)'T$(7);TAB Z;T$(6)'T$(10
);TAB Z;T$(1)'T$(12);TAB Z;T$(5)'T$(2);TAB Z;T$(8)'
6260 LPRINT
6270 LPRINT S$
6280 LPRINT '"WEEK 8:"'D$(15)'T$(2);TAB Z;T$(9)'T$(12);TAB Z;T$(7)'T$(1);TAB Z;T
$(3)'T$(8);TAB Z;T$(5)'T$(4);TAB Z;T$(10)'T$(6);TAB Z;T$(11)'
6290 LPRINT 'D$(16)'T$(5);TAB Z;T$(1)'T$(8);TAB Z;T$(11)'T$(4);TAB Z;T$(2)'T$(7)
;TAB Z;T$(9)'T$(3);TAB Z;T$(6)'T$(12);TAB Z;T$(10)'
6300 GO TO 7300
```

```
7000 LPRINT '"WEEK 1:"'D$(1)'T$(1);TAB Z;T$(2)'T$(3);TAB Z;T$(4)'T$(5);TAB Z;T$(
6)'T$(7);TAB Z;T$(8)'T$(9);TAB Z;T$(10)'T$(11);TAB Z;T$(12)'T$(13);TAB Z;T$(14)
'
7010 LPRINT 'D$(2)'T$(8);TAB Z;T$(5)'T$(7);TAB Z;T$(10)'T$(2);TAB Z;T$(11)'T$(3)
;TAB Z;T$(1)'T$(14);TAB Z;T$(4)'T$(6);TAB Z;T$(13)'T$(9);TAB Z;T$(12)'
7020 LPRINT
7030 LPRINT S$
7040 LPRINT '"WEEK 2:"'D$(3)'T$(7);TAB Z;T$(6)'T$(8);TAB Z;T$(9)'T$(1);TAB Z;T$(
12)'T$(4);TAB Z;T$(2)'T$(13);TAB Z;T$(3)'T$(5);TAB Z;T$(14)'T$(11);TAB Z;T$(10)
'
7050 LPRINT 'D$(4)'T$(12);TAB Z;T$(4)'T$(13);TAB Z;T$(2)'T$(14);TAB Z;T$(7)'T$(9
);TAB Z;T$(6)'T$(8);TAB Z;T$(11)'T$(10);TAB Z;T$(1)'T$(3);TAB Z;T$(5)'
7060 LPRINT
7070 LPRINT S$
7080 LPRINT '"WEEK 3:"'D$(5)'T$(2);TAB Z;T$(9)'T$(1);TAB Z;T$(14)'T$(8);TAB Z;T$
(13)'T$(10);TAB Z;T$(5)'T$(12);TAB Z;T$(7)'T$(3);TAB Z;T$(11)'T$(4);TAB Z;T$(6)
'
7090 LPRINT 'D$(6)'T$(13);TAB Z;T$(10)'T$(5);TAB Z;T$(12)'T$(9);TAB Z;T$(3)'T$(2
);TAB Z;T$(7)'T$(1);TAB Z;T$(6)'T$(4);TAB Z;T$(8)'T$(14);TAB Z;T$(11)'
7100 LPRINT
7110 LPRINT S$
7120 LPRINT '"WEEK 4:"'D$(7)'T$(6);TAB Z;T$(8)'T$(9);TAB Z;T$(7)'T$(4);TAB Z;T$(
10)'T$(1);TAB Z;T$(11)'T$(3);TAB Z;T$(14)'T$(13);TAB Z;T$(5)'T$(12);TAB Z;T$(2)
'
7130 LPRINT 'D$(8)'T$(5);TAB Z;T$(1)'T$(10);TAB Z;T$(6)'T$(3);TAB Z;T$(8)'T$(12)
;TAB Z;T$(13)'T$(11);TAB Z;T$(9)'T$(14);TAB Z;T$(2)'T$(7);TAB Z;T$(4)'
7140 LPRINT
7150 LPRINT S$
7160 LPRINT '"WEEK 5:"'D$(9)'T$(3);TAB Z;T$(12)'T$(11);TAB Z;T$(13)'T$(7);TAB Z;
T$(1)'T$(6);TAB Z;T$(14)'T$(5);TAB Z;T$(2)'T$(9);TAB Z;T$(4)'T$(10);TAB Z;T$(8)
'
7170 LPRINT 'D$(10)'T$(4);TAB Z;T$(11)'T$(14);TAB Z;T$(8)'T$(6);TAB Z;T$(2)'T$(5
);TAB Z;T$(9)'T$(10);TAB Z;T$(12)'T$(7);TAB Z;T$(3)'T$(1);TAB Z;T$(13)'
7180 LPRINT
7190 LPRINT S$
7200 LPRINT '"WEEK 6:"'D$(11)'T$(9);TAB Z;T$(14)'T$(4);TAB Z;T$(1)'T$(11);TAB Z;
T$(5)'T$(8);TAB Z;T$(12)'T$(7);TAB Z;T$(13)'T$(2);TAB Z;T$(10)'T$(6);TAB Z;T$(3
)'
7210 LPRINT 'D$(12)'T$(10);TAB Z;T$(3)'T$(6);TAB Z;T$(11)'T$(12);TAB Z;T$(14)'T$
(13);TAB Z;T$(4)'T$(2);TAB Z;T$(8)'T$(1);TAB Z;T$(9)'T$(5);TAB Z;T$(7)'
7220 LPRINT
7230 LPRINT S$
7240 LPRINT '"WEEK 7:"'D$(13)'T$(11);TAB Z;T$(7)'T$(2);TAB Z;T$(3)'T$(13);TAB Z;
T$(9)'T$(14);TAB Z;T$(10)'T$(4);TAB Z;T$(5)'T$(12);TAB Z;T$(6)'T$(8);TAB Z;T$(1
)'
7250 LPRINT 'D$(14)'T$(14);TAB Z;T$(13)'T$(10);TAB Z;T$(9)'T$(2);TAB Z;T$(1)'T$(
6);TAB Z;T$(5)'T$(8);TAB Z;T$(7)'T$(4);TAB Z;T$(3)'T$(12);TAB Z;T$(11)'
```

```
7260 LPRINT
7270 LPRINT S$
7280 LPRINT '"WEEK 8:"'D$(15)'T$(12);TAB Z;T$(9)'T$(4);TAB Z;T$(14)'T$(5);TAB Z;
T$(8)'T$(11);TAB Z;T$(2)'T$(1);TAB Z;T$(3)'T$(10);TAB Z;T$(7)'T$(13);TAB Z;T$(6
)'
7290 LPRINT 'D$(16)'T$(10);TAB Z;T$(11)'T$(3);TAB Z;T$(13)'T$(6);TAB Z;T$(7)'T$(
12);TAB Z;T$(1)'T$(2);TAB Z;T$(4)'T$(9);TAB Z;T$(8)'T$(14);TAB Z;T$(5)'
7300 LPRINT S$: CLS : PRINT "DO YOU WISH TO SCHEDULE ANY MOREGAMES AT THIS TIME?
": PAUSE 0
7310 IF INKEY$="N" OR INKEY$="n" THEN  CLS : PRINT  FLASH 1;AT 11,0;"It was a pl
easure to serve you!": STOP
7320 CLS : PRINT "SAME GAME DATES?": PAUSE 0
7330 IF INKEY$="Y" OR INKEY$="y" THEN  GO TO 140
7340 GO TO 90
7350 STOP
9000 SAVE "SCHEDULER" LINE 10
```

POLYNOMIAL CURVE FIT PROGRAM

by Stan Lemke

INTRODUCTION

The following Polynomial Curve Fit Program for the Timex Sinclair TS2068 is a powerful mathematical tool for students, engineers, and everyone interested in or involved with mathematics.

You can INPUT X-Y coordinate data and plot the points, or you can have the PCF (Polynomial Curve Fit Program) plot the data with straight line segments connecting the points, you can have the PCF connect the points with a piece-wise continuous (spline) curve, or the PCF can curve fit up to a 10th order polynomial to the data and plot it with the data points. You can view the polynomial equation used in the curve fit (note: the entire equation and not just the polynomial coefficients!). You can compare the input data values with the curve fit values at the same "X" coordinate points. You can evaluate the polynomial at "X" coordinate points beyond that used to define the curve.

Polynomial curve fit analysis is only one function available to the PCF. Instead of inputting coordinate data, you can input an entire equation and have the PCF evaluate and plot it! This equation can be a polynomial, a trigonometric function, logarithmic function, or a combination of all of these.

HOW TO USE THE PROGRAM

The Polynomial Curve Fit (PCF) Program will load and auto-run. The PCF will give a selection of 2 modes. Press 1 if you wish to enter X-Y coordinate data. Press 2 if you wish to input an equation.

After the data has been entered (or the equation has been entered), you will be prompted with a menu containing 6 choices. Press the number key (1 to 6) associated with the desired option.

Press 1 -- Display the Polynomial (equation)
2 -- Compare the Input/Curve data
3 -- Evaluate Curve at "X"
4 -- Plot Input/Curve data
5 -- Next Case
6 -- End Job

Press 1 to display the Polynomial Curve Fit equation (if you entered X-Y coordinate data), or the equation you input if that was your choice.

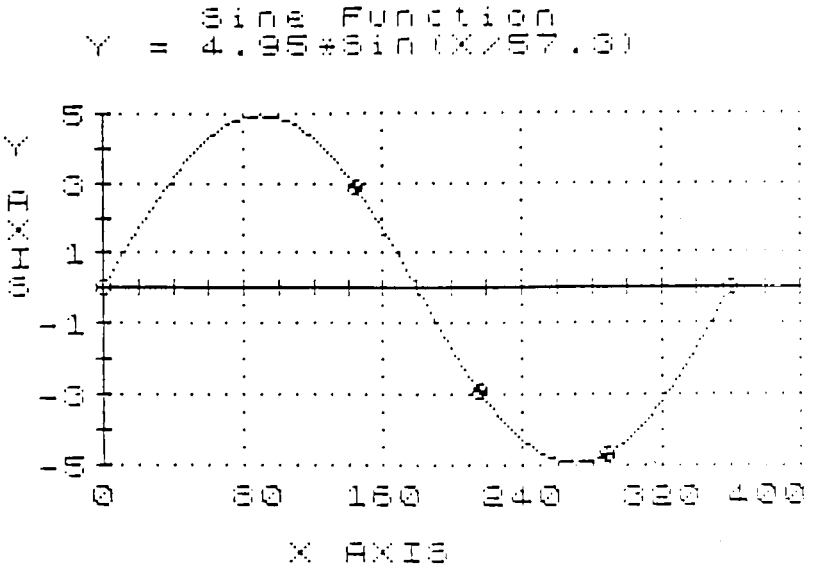
Press 2 to compare and display the input X-Y coordinate data and the curve fit (or input) equation data. Note: if you input an equation, you also select a minimum and maximum "X" over which to evaluate the equation. The PCF selects three intermediate points for a total of five points to use as input data points.

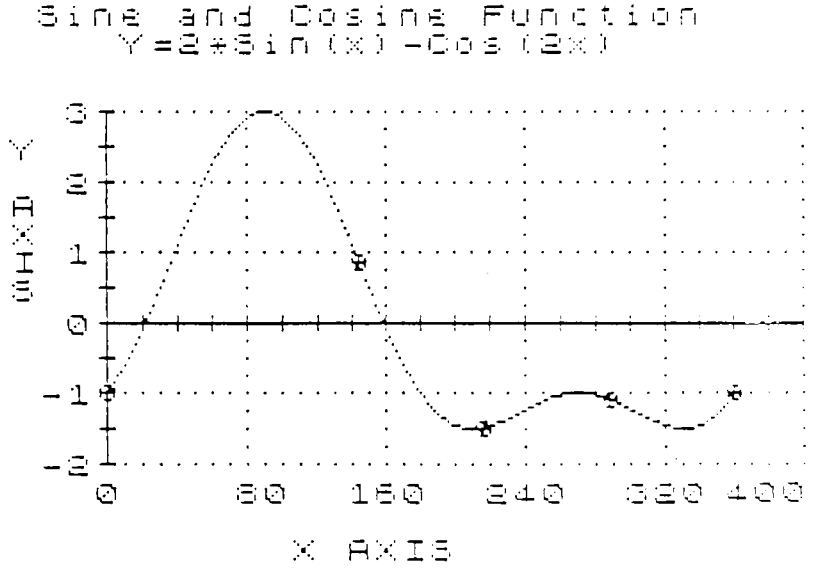
Press 3 to evaluate the polynomial (or input) equation at user-defined "X" values.

Press 4 to plot the X-Y coordinate data and curves. You will be given another menu containing five choices.

Press 1 -- Plot Input Data Only
2 -- Straight Line Segments
3 -- Curves Line Segments (Spline Curve)
4 -- Polynomial Curve Fit (or Input Equation)
5 -- No plots, Return

Simply press the number key associated with the desired option. If you select to plot a curve, you will be prompted with the current page title, sub-title, x-axis title, and y-axis titles. Type in the desired title, or just press ENTER to retain the present title. When the page/curve is complete, you will be prompte with: COPY? Press "Y" or "N". The page will be displayed until you press a key. If you press "Y", you will get a printer (TS2040 thermal





printer) COPY of the screen. When the COPY is complete, or if you pressed anything except "Y", the last (plot type) menu will be displayed again.

Press 5 to start a new case. You will start over and can select to input X-Y coordinate data, or input an equation. However, if in the previous case you either input an equation or have one created in the curve fit function, then you can return and use it again by selecting the equation input mode. When prompted with "Input your equation...", just press ENTER. The PCF will use the old equation in this case. Next you will be asked to define the "X" limits, min. and max. to be evaluated. Type in the desired values and press ENTER.

Press 6 to terminate the job.

INPUT X-Y COORDINATE DATA

When you select this option, the PCF will prompt you to input the desired curve fit order. Type a number from 1 to 10, and press ENTER. If you do not wish to have a polynomial fit to your data, input "0".

Next, the PCF will ask you how many data points you wish to enter. The PCF will allow up to 50 coordinate pairs, and will prompt you for the number of data pairs you indicate. Note: the number of data points must be at least 1 greater than the order of the polynomial curve fit requested.

The PCF will begin to curve fit your data when the last data point is entered. A 2nd or 3rd order polynomial will take only a few seconds to solve. Higher order equations will take several minutes! When the curve fit is completed, the PCF will go to the six option menu.

INPUT AN EQUATION

When you select this option, the PCF will prompt you for an equation. If you have previously input an equation, or have one created by the curve-fit function, you can select this curve by pressing ENTER.

Otherwise, INPUT your equation by typing in the equation as a function of "x" (lower case!!), and press ENTER.

```
Example 1: Y = 3.5 + x
           Input:    3.5 + x  (ENTER)
Example 2: Y = 2*Sin(x/57.3)
           Input:    2*SIN(x/57.3)   (ENTER)
```

You will be prompted for a minimum "x" value (evaluate from) and a maximum "x" value (evaluate to). If you input a polynomial equation, or any equation where you raise "x" to a power, you cannot use the "arrow" function like x^2 with negative values of "x". Instead use x*x. Also, to use the Sine function, use the keyboard "SIN" command (do not spell it).

FULL SIZE PRINTER MODIFICATION

The Polynomial Curve Fit Program is written to use the TS2040 type printer. However, it is very easy to modify the program to use the full size printer.

Step 1: Load the PCF program and press the CAPS SHIFT + BREAK keys to break the program.

Step 2: Type the following lines:

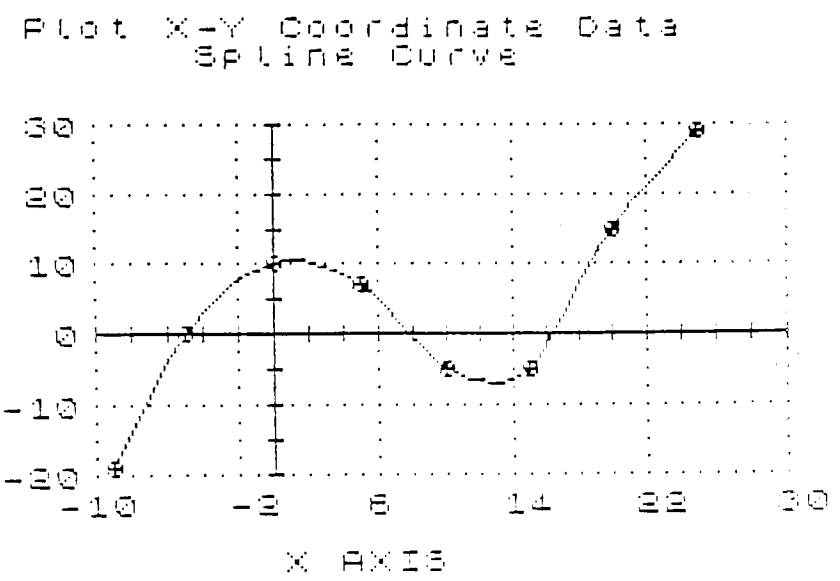
```
  40 LOAD "" CODE
3400 RANDOMIZE USR 23296
```

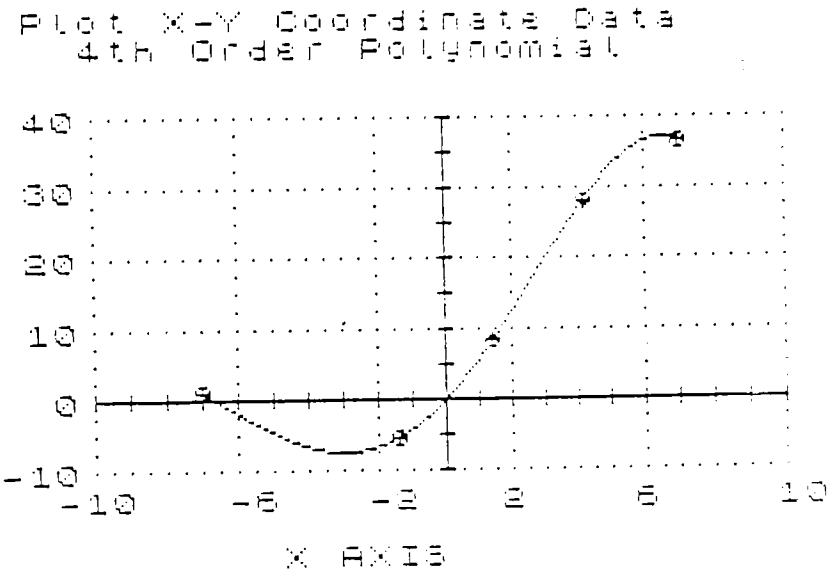
Step 3: Save the new PCF program on a new tape, and type GOTO 9999 (and press ENTER).

Step 4: Load a copy of the graphics screen copy code supplied with your printer interface into your TS2068, then save it to the tape including and following the new PCF program you just created. When PCF runs, the first thing that it will do, will be to read this code into memory. These instructions assume that your printer code is located in RAM at location 23296. If it is not located here, additional program modifications beyond the scope of this documentation is required.

```
  1 REM  Polynomial Curve Fit
 10 REM **************************
 20 REM by S D Lemke  1984
          2144 White Oak
            Wichita, Ks 67207
 30 REM **************************
 40 REM
 45 CLEAR 65229: GO SUB 5100
 50 PAPER 1: BORDER 1: INK 9: C
LS : FOR i=0 TO 7: POKE USR "a"+
i,0: NEXT i: POKE USR "a"+1,255
 65 LET ix=0: LET num=0: LET e$
="0"
 70 LET le=0: LET x$="X AXIS":
LET z$="Y AXIS": LET m$="Main Ti
tle": LET n$="Sub-Title"
 80 DEF FN A(X)=INT (X*1000)/10
00
 85 DEF FN b(a,b,c,d,e)=(a-b)/(
c-d)*(e-d)+d
 90 DEF FN Z(Q,R)=Q^R
100 DIM v(4,201): DIM c(10): DI
M f(50,10): DIM t(10,50): DIM s(
10,11)
105 PRINT AT 5,10;"Polynomial";
';TAB 7;"Curve Fit Program";';TA
B 7;"????????????????????": GO TO 1
000 :REM  "?" = UDG "A"
110 INPUT "Order of polynomial
(<=10) ? ";norder: IF norder>=0
AND norder<=10 THEN  PRINT AT 8,
7;"Polynomial order = ";norder
115 IF norder<0 OR norder>=11 T
HEN  PRINT AT 12,9; FLASH 1;"Inv
alid ""Order""": PAUSE 120: PRIN
T AT 12,9;"                  ": GO
TO 110
120 INPUT "Number of points on
the curve (<=50)? ";num: IF num>
=norder+1 AND num<=50 THEN  GO T
O 128
125 PRINT AT 12,9; FLASH 1;"Inv
alid ""Number""": PAUSE 120: PRI
NT AT 12,9;"                  ": G
O TO 120
128 PRINT ';TAB 7;"There are ";
num;" points.";':
130 PRINT AT 12,10;"Input Data"
;';TAB 10;"??????????": FOR i=1
TO num: INPUT " X = ? ";v(1,i);"
      Y = ? ";v(3,i): PRINT TAB
5;i;TAB 9;"X = ";v(1,i);TAB 22;
"Y = ";v(3,i): POKE 23692,51: NE
XT i: REM   NOTE..." ? " are the
 User Defined Graphics AAA
140 IF norder>=1 THEN  PRINT #0
;AT 0,10;"Working": GO TO 200
150 CLS
160 PRINT AT 7,1;"Select an opt
ion: Press 1 to 6": PRINT AT 9,3
;"1 -- Display Polynomial Eq.":
PRINT AT 10,3;"2 -- Compare Inpu
t/Curve Data": PRINT AT 11,3;"3
-- Evaluate Curve at ""X"" ": PR
INT AT 12,3;"4 -- Plot Input/Cur
ve Data": PRINT AT 13,3;"5 -- Ne
xt Case": PRINT AT 14,3;"6 -- En
d Job": PAUSE 0: LET i$=INKEY$:
IF CODE i$<49 OR CODE i$>54 THEN
  GO TO 160
161 IF norder=0 AND VAL i$<4 TH
EN  GO TO 160
165 IF i$="1" THEN  CLS : PRINT
 AT 3,5;"Polynomial Curve Fit";'
;TAB 9;"Order = ";norder: PRINT
AT 7,0;"Y = "+e$: GO SUB 3380: G
O TO 150
170 IF i$="2" THEN  GO TO 320
175 IF i$="3" THEN  CLS : PRINT
 AT 1,0;"Y = "+e$: INPUT "X = ";
x: LET le=LEN e$/30+3: PRINT AT
le,0;"X = ";x;TAB 15;"Y = ";VAL
e$
176 IF i$="3" THEN  PRINT AT 21
,5; FLASH 1;"Press Enter to Cont
inue": PAUSE 0: PRINT AT 21,5;"
                    ": IF INK
EY$<>CHR$ 13 THEN  GO TO 150
177 IF i$="3" AND le>=20 THEN
CLS : PRINT AT 1,0;"Y = "+e$: LE
T le=LEN e$/30+2
```

```
178 IF i$="3" THEN  INPUT "X =
";x: LET le=le+1: PRINT AT le,0;
"X = ";x;TAB 15;"Y = ";VAL e$: G
O TO 176
180 IF i$="4" THEN  CLS : GO TO
2000
185 IF i$="5" THEN  CLS : GO TO
105
190 STOP
200 REM     Begin Curve Fit

210 GO SUB 3200: LET x1=minx: L
ET x2=maxx: LET y1=miny: LET y2=
maxy: LET dx=x2-x1: LET dy=y2-y1
: LET s=dy/dx: FOR i=1 TO num: L
ET v(2,i)=v(1,i)-x1: LET v(4,i)=
v(3,i)-y1-s*v(2,i): NEXT i: LET
v(4,num)=0
220 REM   Calculate "F" Matrix

230 FOR j=1 TO num: FOR i=1 TO
norder: LET f(j,i)=FN z(v(2,j)/d
x,i): LET t(i,j)=f(j,i): NEXT i:
NEXT j
240 REM Simulataneous Equations

250 FOR i=1 TO norder: FOR j=1
TO norder: LET s(i,j)=0: FOR k=1
TO num: LET s(i,j)=s(i,j)+t(i,k
)*f(k,j): NEXT k: NEXT j: NEXT i
260 REM Calc. Column Constants

270 FOR i=1 TO norder: LET c(i)
=0: FOR k=1 TO num: LET c(i)=c(i
)+t(i,k)*v(4,k): NEXT k: NEXT i:
FOR i=1 TO norder: LET s(i,nord
er+1)=c(i): NEXT i
280 REM Solve system of Sim. Eq
290 LET mp1=norder+1: FOR j=2 T
O mp1: LET s(1,j)=s(1,j)/s(1,1):
NEXT j: FOR i=2 TO norder: FOR
h=i TO norder: LET sum=0: LET jm
1=i-1: FOR k=1 TO jm1: LET sum=s
um+s(h,k)*s(k,i): NEXT k: LET s(
h,i)=s(h,i)-sum: NEXT h: LET ip1
=i+1: FOR j=ip1 TO mp1: LET sum=
0: LET im1=i-1: FOR k=1 TO im1:
LET sum=sum+s(i,k)*s(k,j): NEXT
k: LET s(i,j)=(s(i,j)-sum)/s(i,i
): NEXT j: NEXT i: LET c(norder)
=s(norder,mp1): LET l=norder-1:
FOR n=1 TO l: LET sum=0: LET i=n
order-n: LET ip1=i+1: FOR j=ip1
TO norder: LET sum=sum+s(i,j)*c(
j): NEXT j: LET c(i)=s(i,mp1)-su
m: NEXT n: LET c(1)=c(1)+s*dx
300 REM Calculate Output Matrix

310 FOR i=1 TO num: LET sum=0:
FOR j=1 TO norder: LET sum=sum+c
(j)*FN z(v(2,i)/dx,j): NEXT j: L
ET v(4,i)=sum: NEXT i: FOR i=1 T
O num: LET v(2,i)=v(2,i)+x1: LET
v(4,i)=v(4,i)+y1: NEXT i: GO TO
345
320 REM Print Input/Output Data

325 LET i1=1
330 CLS : PRINT AT 0,0;"X-in
Y-in     X-out     Y-out": PRINT A
T 1,0;;"????     ????     ?????
?????"
335 FOR i=i1 TO num: LET x=v(1,
i): LET v(2,i)=x: LET v(4,i)=VAL
e$: PRINT FN a(v(1,i));TAB 8;FN
a(v(3,i));TAB 16;FN a(v(2,i));T
AB 24;FN a(v(4,i)): IF i=19 OR i
=39 THEN  LET i1=i+1: GO SUB 338
0: GO TO 330
340 NEXT i: GO SUB 3380: GO TO
150
345 REM  PRINT Create Polynomia
l
350 LET y1=FN a(y1): LET e$=" "
360 FOR i=1 TO norder: IF FN a(
c(i))<>0 THEN  LET e$=e$+STR$ FN
a(c(i))+"*((X"
370 IF FN a(c(i))<>0 AND FN a(x
1)<>0 THEN  LET e$=e$+"-"+STR$ F
N a(x1)
380 IF FN a(c(i))<>0 THEN  LET
e$=e$+")/"+STR$ FN a(dx)+")^"+ST
R$ i+" + "
390 NEXT i: LET e$=e$+STR$ y1:
GO TO 150
395 REM   Scaling Subroutine

400 DATA .001,.002,.005,.01,.02
,.05,.1,.2,.5,1,2,5,10,20,50,100
,200,300,400,500,600,700,800,900
,1000,1100,1200,1300,1400,1500,1
600,1700,1800,1900,2000,2100,220
0,2300,2400,2500,2600,2700,2800,
2900,3000,5000,10000,20000,10000
0
```





```
 410 DIM s(49): RESTORE 400
 420 FOR i=1 TO 49: READ s(i): N
EXT i
 430 IF min=0 THEN  GO TO 500
 440 LET amin=ABS min: GO SUB 62
0: IF i1=1 THEN  GO TO 500
 450 IF min<0 THEN  GO TO 480
 460 REM  IF i1=3 OR i1=6 OR i1=
9 OR i1=12 OR i1=15 THEN  LET i1
=i1-1
 470 LET min=s(i1): GO TO 510
 480 REM  IF i2=3 OR i2=6 OR i2=
9 OR i2=12 OR i2=15 THEN  LET i2
=i2+1
 490 LET min=-s(i2): GO TO 510
 500 LET min=0
 510 LET d1=(max-min)/a1: LET am
in=d1: GO SUB 620
 520 REM  IF i2=3 OR i2=6 OR i2=
9 OR i2=12 OR i2=15 THEN  LET i2
=i2+1
 530 IF i2>=18 AND i2<=20 THEN
LET i2=20
 540 IF i2>=21 AND i2<=25 THEN
LET i2=25
 550 IF i2>=26 AND i2<=30 THEN
LET i2=30
 560 IF i2>=31 AND i2<=35 THEN
LET i2=35
 570 IF i2>=36 AND i2<=40 THEN
LET i2=40
 580 IF i2>=41 AND i2<=46 THEN
LET i2=46
 590 IF i2<=49 THEN  LET d1=s(i2
)
 600 IF ABS min<=d1 THEN  LET mi
n=0
 610 RETURN
 620 FOR i=2 TO 49
 630 IF s(i)>amin THEN  GO TO 66
0
 640 NEXT i
 650 LET i=50
 660 LET i2=i: LET i1=i-1: RETUR
N
1000 PRINT AT 10,0;"Select an Op
tion. Press 1 or 2.": PRINT AT 1
4,0;"Press  1  to INPUT X and Y
Data        2  to INPUT an EQUAT
ION"
1010 PAUSE 0: LET i$=INKEY$: IF
i$<"1" OR i$>"2" THEN  GO TO 101
0
1020 PRINT AT 10,0;TAB 31;" ";AT
 14,0;TAB 31;" ";TAB 31;" ": IF
i$="1" THEN  GO TO 110
1030 PRINT AT 10,0;"Input your E
quation:   Y = ": INPUT "Y = ";g
$: IF LEN g$<>0 THEN  LET e$=g$:
 PRINT AT 11,7;e$: LET norder=1:
 LET num=5: INPUT "Evaluate Equa
tion from X = ";minx: INPUT "Eva
luate Equation to X = ";maxx: PR
INT #0;AT 0,12;"WORKING": LET v(
1,1)=minx: LET v(1,5)=maxx: FOR
i=2 TO 4: LET v(1,i)=minx+i*(max
x-minx)/5: NEXT i: FOR i=1 TO 5:
 LET x=v(1,i): LET v(3,i)=VAL e$
: NEXT i: GO SUB 3200: CLS : GO
TO 160
1040 PRINT AT 11,7;e$: LET norde
r=1: LET num=5: INPUT "Evaluate
Equation from X = ";minx: INPUT
"Evaluate Equation to X = ";maxx
: PRINT #0;AT 0,12;"WORKING": LE
T v(1,1)=minx: LET v(1,5)=maxx:
FOR i=2 TO 4: LET v(1,i)=minx+i*
(maxx-minx)/5: NEXT i: FOR i=1 T
O 5: LET x=v(1,i): LET v(3,i)=VA
L e$: NEXT i: GO SUB 3200: CLS :
 GO TO 160
2000 REM Select type of plot
2010 PRINT AT 7,1;"Select an opt
ion: Press 1 to 5": PRINT AT 9,3
;"1 -- Plot INPUT data only
  2 -- Straight line segments
  3 -- Curved line segments
  4 -- Polynomial Curve Fit
  5 -- No plots, RETURN": PAUSE
0: LET i$=INKEY$: IF i$<"1" OR i
$>"5" THEN  GO TO 2010
2020 IF i$="5" THEN  GO TO 150
2030 IF i$="4" THEN  GO TO 3280
2040 GO SUB 6280
2050 IF i$="3" THEN  PRINT #0;AT
 0,12;"WORKING": GO SUB 6500
2060 IF i$="2" THEN  GO SUB 7500
2070 GO SUB 3380: GO TO 150
3000 CLS : PRINT AT 2,0;"PAGE TI
TLE:  "; FLASH 1;m$
3010 INPUT "New Title is: ";t$:
IF LEN t$<>0 THEN  LET m$=t$
3020 PRINT AT 2,13;TAB 31;" ";AT
 2,13;m$
3030 PRINT AT 4,0;"SUB TITLE:
"; FLASH 1;n$
3040 INPUT "New Title is: ";t$:
IF LEN t$<>0 THEN  LET n$=t$
3050 PRINT AT 4,13;TAB 31;" ";AT
 4,13;n$
3100 PRINT AT 6,0;"X-AXIS TITLE:
  "; FLASH 1;x$
3110 INPUT "New Title is: ";t$:
IF LEN t$<>0 THEN  LET x$=t$
3120 PRINT AT 6,15;TAB 31;" ";AT
 6,15;x$
3130 PRINT AT 8,0;"Y-AXIS TITLE:
  "; FLASH 1;z$
3140 INPUT "New Title is: ";t$:
IF LEN t$<>0 THEN  LET z$=t$
3150 PRINT AT 8,15;TAB 31;" ";AT
 8,15;z$
3160 PRINT AT 12,12;"WORKING": R
ETURN
3200 REM       Plot Subroutine

3210 LET maxx=v(1,1): LET minx=v
(1,1): LET maxy=v(3,1): LET miny
=v(3,1): FOR i=1 TO num
3230 IF v(1,i)>maxx THEN  LET ma
xx=v(1,i)
3240 IF v(3,i)>maxy THEN  LET ma
xy=v(3,i)
3250 IF v(1,i)<minx THEN  LET mi
nx=v(1,i)
3260 IF v(3,i)<miny THEN  LET mi
ny=v(3,i)
3270 NEXT i: RETURN
3280 GO SUB 3000: IF norder=0 TH
EN  GO SUB 3200
3285 LET min=minx: LET max=maxx:
 LET a1=20: GO SUB 400: LET x0=m
in: LET dx=d1: LET xn=0: IF x0<0
 THEN  LET xn=-x0/dx
3290 LET min=miny: LET max=maxy:
 LET a1=10: GO SUB 400: LET y0=m
in: LET dy=d1: LET yn=0: IF y0<0
 THEN  LET yn=-y0/dy
3300 IF xn>20 THEN  LET xn=20
3310 IF yn>10 THEN  LET yn=10
3320 CLS : LET xo=44: LET yo=28:
 PLOT xo+xn*10,yo: DRAW 0,100: F
OR i=0 TO 10: PLOT xo-2+xn*10,yo
```

```
+10*i: DRAW 4,0: NEXT i: PLOT xo
,yo+yn*10: DRAW 200,0: FOR i=0 T
O 20: PLOT xo+10*i,yo-2+yn*10: D
RAW 0,4: NEXT i
3325 FOR i=0 TO 200 STEP 5: PLOT
 xo+i,yo: PLOT xo+i,yo+20: PLOT
xo+i,yo+40: PLOT xo+i,yo+60: PLO
T xo+i,yo+80: PLOT xo+i,yo+100:
NEXT i: FOR i=0 TO 100 STEP 5: P
LOT xo,yo+i: PLOT xo+40,yo+i: PL
OT xo+80,yo+i: PLOT xo+120,yo+i:
 PLOT xo+160,yo+i: PLOT xo+200,y
o+i: NEXT i
3327 LET lz=LEN z$: IF lz>10 THE
N  LET lz=10
3328 LET top=11-INT (lz/2)
3330 PRINT AT 2,(15-LEN m$/2);m$
;AT 3,(15-LEN n$/2);n$: LET y$=z
$: FOR i=1 TO lz: PRINT AT top+i
,0;y$(i): NEXT i: FOR i=1 TO 6:
LET v$=STR$ (2*(i-1)*dy+y0): LET
 lv=LEN v$: FOR h=1 TO lv: LET a
x=36-8*lv+8*h: LET ch=8*(CODE v$
(h)-32)+15616: FOR o=0 TO 7: LET
 y=31+20*(i-1)-o: GO SUB 5000: P
OKE mloc,PEEK (ch+o): NEXT o: NE
XT h: NEXT i
3333 LET lx=LEN x$: IF lx>10 THE
N  LET lx=10
3335 PRINT AT 21,15-INT (lx/2);x
$: FOR i=1 TO 5: LET v$=STR$ (4*
(i-1)*dx+x0): LET lv=LEN v$: FOR
 h=1 TO lv: LET ax=48-8*(lv/2)+8
*(h-1)+40*(i-1): LET ch=8*(CODE
v$(h)-32)+15616: FOR o=0 TO 7: L
ET y=23-o: GO SUB 5000: POKE mlo
c,PEEK (ch+o): NEXT o: NEXT h: N
EXT i
3340 LET v$=STR$ (20*dx+x0): LET
 lv=LEN v$: IF lv<4 THEN  LET i=
6: FOR h=1 TO lv: LET ax=48-8*(1
v/2)+8*(h-1)+40*(i-1): LET ch=8*
(CODE v$(h)-32)+15616: FOR o=0 T
O 7: LET y=23-o: GO SUB 5000: PO
KE mloc,PEEK (ch+o): NEXT o: NEX
T h
3350 REM       Plot the Data

3355 FOR i=1 TO num: LET xp=((v(
1,i)-x0)/dx)*10+xo: LET yp=((v(3
,i)-y0)/dy)*10+yo: PLOT xp,yp: D
RAW -2,0: DRAW 4,0: DRAW -2,0: D
RAW 0,2: DRAW 0,-4: CIRCLE xp,yp
,2: NEXT i
3360 PRINT #0;AT 0,12;"WORKING":
 IF norder>0 THEN  LET ix=(maxx-
minx)*10/dx: FOR i=1 TO ix: LET
x=((i-1)*dx)/10+minx: LET v(2,i)
=x: LET v(4,i)=VAL e$: PLOT INT
(((v(2,i)-x0)/dx)*10+xo+.5),INT
(((v(4,i)-y0)/dy)*10+yo+.5): NEX
T i
3370 GO SUB 3380: GO TO 150
3380 PRINT #0;AT 0,0;"
  Copy?                         Pres
s ""Y"" or ""N"" "
3390 PAUSE 0: LET y$=INKEY$: IF
y$="n" OR y$="N" THEN  GO TO 341
0
3400 COPY
3410 PRINT #0;AT 0,0;"

                              ": RETURN
5000 POKE 65260,ax: POKE 65261,y
: RANDOMIZE USR 65230: LET mloc=
256*PEEK 65263+PEEK 65262: RETUR
N
5110 DATA 237,75,236,254,197,17,
0,0,213,213,17,87,0,213,205,0,98
,34,238,254,50,240,254,193,201
5120 RESTORE 5110: FOR i=65230 T
O 65254: READ a: POKE i,a: NEXT
i: RETURN
6280 GO SUB 3000: GO SUB 3200
6285 LET min=minx: LET max=maxx:
 LET al=20: GO SUB 400: LET x0=m
in: LET dx=dl: LET xn=0: IF x0<0
 THEN  LET xn=-x0/dx
6290 LET min=miny: LET max=maxy:
 LET al=10: GO SUB 400: LET y0=m
in: LET dy=dl: LET yn=0: IF y0<0
 THEN  LET yn=-y0/dy
6300 IF xn>20 THEN  LET xn=20
6310 IF yn>10 THEN  LET yn=10
6320 CLS : LET xo=44: LET yo=28:
 PLOT xo+xn*10,yo: DRAW 0,100: F
OR i=0 TO 10: PLOT xo-2+xn*10,yo
+10*i: DRAW 4,0: NEXT i: PLOT xo
,yo+yn*10: DRAW 200,0: FOR i=0 T
O 20: PLOT xo+10*i,yo-2+yn*10: D
RAW 0,4: NEXT i
6325 FOR i=0 TO 200 STEP 5: PLOT
 xo+i,yo: PLOT xo+i,yo+20: PLOT
xo+i,yo+40: PLOT xo+i,yo+60: PLO
T xo+i,yo+80: PLOT xo+i,yo+100:
NEXT i: FOR i=0 TO 100 STEP 5: P
LOT xo,yo+i: PLOT xo+40,yo+i: PL
OT xo+80,yo+i: PLOT xo+120,yo+i:
 PLOT xo+160,yo+i: PLOT xo+200,y
o+i: NEXT i
6327 LET lz=LEN z$: IF lz>10 THE
N  LET lz=10
6328 LET top=11-INT (lz/2)
6330 PRINT AT 2,(15-LEN m$/2);m$
;AT 3,(15-LEN n$/2);n$: LET y$=z
$: FOR i=1 TO lz: PRINT AT top+i
,0;y$(i): NEXT i: FOR i=1 TO 6:
LET v$=STR$ (2*(i-1)*dy+y0): LET
 lv=LEN v$: FOR h=1 TO lv: LET a
x=36-8*lv+8*h: LET ch=8*(CODE v$
(h)-32)+15616: FOR o=0 TO 7: LET
 y=31+20*(i-1)-o: GO SUB 5000: P
OKE mloc,PEEK (ch+o): NEXT o: NE
XT h: NEXT i
6333 LET lx=LEN x$: IF lx>10 THE
N  LET lx=10
6335 PRINT AT 21,15-INT (lx/2);x
$: FOR i=1 TO 5: LET v$=STR$ (4*
(i-1)*dx+x0): LET lv=LEN v$: FOR
 h=1 TO lv: LET ax=48-8*(lv/2)+8
*(h-1)+40*(i-1): LET ch=8*(CODE
v$(h)-32)+15616: FOR o=0 TO 7: L
ET y=23-o: GO SUB 5000: POKE mlo
c,PEEK (ch+o): NEXT o: NEXT h: N
EXT i
6340 LET v$=STR$ (20*dx+x0): LET
 lv=LEN v$: IF lv<4 THEN  LET i=
6: FOR h=1 TO lv: LET ax=48-8*(1
v/2)+8*(h-1)+40*(i-1): LET ch=8*
(CODE v$(h)-32)+15616: FOR o=0 T
O 7: LET y=23-o: GO SUB 5000: PO
KE mloc,PEEK (ch+o): NEXT o: NEX
T h
6370 FOR i=1 TO num: LET xp=((v(
1,i)-x0)/dx)*10+xo: LET yp=((v(3
,i)-y0)/dy)*10+yo: PLOT xp,yp: D
RAW -2,0: DRAW 4,0: DRAW -2,0: D
RAW 0,2: DRAW 0,-4: CIRCLE xp,yp
,2: NEXT i
6380 RETURN
6499 REM Spline line segments
6500 IF num<=1 THEN  RETURN
6505 IF minx=maxx THEN  RETURN
6510 LET ix=INT ((maxx-minx)*10/
dx/3): LET dxx=(maxx-minx)/(ix-1
```

```
): FOR i=1 TO ix: LET v(2,i)=dxx
*(i-1)+minx: NEXT i
6520 LET in=0: FOR n=1 TO ix: IF
 num=1 THEN  LET v(4,n)=v(3,1):
GO TO 6820
6530 IF num=2 THEN  LET v(4,n)=F
N b(v(3,2),v(3,1),v(1,2),v(1,1),
v(2,n)): GO TO 6820
6540 IF num>2 THEN  GO TO 6580
6550 IF v(1,2)<v(2,n) THEN  LET
v(4,n)=FN b(v(3,3),v(3,2),v(1,3)
,v(1,2),v(2,n)): GO TO 6820
6560 IF v(1,2)=v(2,n) THEN  LET
v(4,n)=v(3,2): GO TO 6820
6570 LET v(4,n)=FN b(v(3,2),v(3,
1),v(1,3),v(1,2),v(2,n)): GO TO
6820
6580 LET js=1
6590 IF v(1,js)=v(2,n) THEN  LET
 v(4,n)=v(3,js): GO TO 6820
6600 IF v(1,js)>v(2,n) THEN  GO
TO 6680
6610 LET js=js+1: IF js<=num THE
N  GO TO 6590
6620 LET v(4,n)=FN b(v(3,num),v(
3,num-1),v(1,num),v(1,num-1),v(2
,n)): GO TO 6820
6630 IF js=1 THEN  LET ks=3: LET
 jj=1: GO TO 6710
6640 IF js>num THEN  GO TO 6620
6650 IF js=num THEN  LET ks=num-
1: LET jj=2: GO TO 6710
6660 IF js=in THEN  GO TO 6760
6670 LET jj=3: LET ks=js: GO TO
6710
6680 IF js=2 THEN  LET ks=3: LET
 jj=1: GO TO 6710
6690 IF js>2 THEN  GO TO 6640
6700 LET v(4,n)=v(3,2): GO TO 68
20
6710 IF ks<3 THEN  LET ks=3
6720 DIM e(2): DIM b(2): DIM d(2
): DIM g(2)
6730 FOR m=1 TO 2: LET x1=v(1,ks
-1)-v(1,ks): LET x2=v(1,ks)-v(1,
ks-2): LET x3=v(1,ks-2)-v(1,ks-1
)
6735 LET y1=v(3,ks-1)-v(3,ks): L
ET y2=v(3,ks)-v(3,ks-2): LET y3=
v(3,ks-2)-v(3,ks-1)
6740 LET xx1=v(1,ks-2)*v(1,ks-2)
: LET xx2=v(1,ks-1)*v(1,ks-1): L
ET xx3=v(1,ks)*v(1,ks)
6745 LET d(m)=xx1*x1+xx2*x2+xx3*
x3: LET e(m)=(v(3,ks-2)*x1+v(3,k
s-1)*x2+v(3,ks)*x3)/d(m): LET b(
m)=(xx1*y1+xx2*y2+xx3*y3)/d(m):
LET g(m)=v(3,ks-2)-e(m)*xx1-b(m)
*v(1,ks-2): IF ks<>num THEN  LET
 ks=ks+1
6750 NEXT m
6760 LET p1=v(2,n)*(e(1)*v(2,n)+
b(1))+g(1): REM  LET slp1=2*e(1)
*v(2,n)+b(1): LET slp2=2*e(2)*v(
2,n)+b(2): IF js=1 THEN  LET slp
3=(v(3,2)-v(3,1))/(v(1,2)-v(1,1)
)
6765 REM  IF js>=2 THEN  LET slp
3=(v(3,js)-v(3,js-1))/(v(1,js)-v
(1,js-1))
6770 IF jj=1 THEN  LET p2=p1: LE
T al=(v(2,n)-v(1,1))/(v(1,2)-v(1
,1)): LET ss=al*v(3,2)+(1-al)*v(
3,1): LET p1=ss+(p2-ss)/2: GO TO
 6800: REM  LET slp2=slp1: LET s
lp1=slp3+(slp2-slp3)/2: GO TO 68
00
6780 IF jj=2 THEN  LET p2=v(2,n)
*(e(2)*v(2,n)+b(2))+g(2): LET p1
=p2: LET al=(v(2,n)-v(1,num-1))/
(v(1,num)-v(1,num-1)): LET ss=al
*v(3,num)+(1-al)*v(3,num-1): LET
 p2=ss+(p1-ss)/2: GO TO 6800: RE
M  LET slp1=slp2: LET slp2=slp3+
(slp1-slp3)/2: GO TO 6800
6790 IF jj=3 THEN  LET p2=v(2,n)
*(e(2)*v(2,n)+b(2))+g(2): LET al
=(v(2,n)-v(1,js-1))/(v(1,js)-v(1
,js-1)): LET ss=al*v(3,js)+(1-al
)*v(3,js-1)
6800 LET e1=ABS (p1-ss): LET e2=
ABS (p2-ss): LET in=js: IF (e1+e
2)<=0.00001 THEN  LET v(4,n)=ss:
 GO TO 6820
6810 LET bt=(e1*al)/(e1*al+(1-al
)*e2): LET v(4,n)=bt*p2+(1-bt)*p
1
7465 PLOT INT (((v(2,n)-x0)/dx)*
10+xo+.5),INT (((v(4,n)-y0)/dy)*
10+yo+.5): NEXT n
7470 LET xp=INT (((v(2,1)-x0)/dx
)*10+xo+.5): LET yp=INT (((v(4,1
)-y0)/dy)*10+yo+.5): PLOT xp,yp
7480 FOR i=2 TO ix: LET xp2=INT
(((v(2,i)-x0)/dx)*10+xo+.5): LET
 yp2=INT (((v(4,i)-y0)/dy)*10+yo
+.5): PLOT xp,yp: DRAW xp2-xp,yp
2-yp: LET xp=xp2: LET yp=yp2: NE
XT i: RETURN
7499 REM Straight line segments
7500 LET xp=((v(1,1)-x0)/dx)*10+
xo: LET yp=((v(3,1)-y0)/dy)*10+y
o: PLOT xp,yp
7510 FOR i=2 TO num: LET xp2=((v
(1,i)-x0)/dx)*10+xo: LET yp2=((v
(3,i)-y0)/dy)*10+yo: PLOT xp,yp:
 DRAW (xp2-xp),(yp2-yp): LET xp=
xp2: LET yp=yp2: NEXT i: RETURN
9998 STOP
9999 CLEAR : SAVE "C-Fit" LINE 1
```

DICE II

by John Deering

```
   3 CLS : BORDER NOT PI
   5 PAPER NOT PI: INK VAL "7":
BORDER NOT PI
  10 DIM D$(6,9)
  20 LET D$(1)="****0****"
  30 LET D$(2)="***0*0***"
  40 LET D$(3)="0***0***0"
  50 LET D$(4)="0*0***0*0"
  60 LET D$(5)="0*0*0*0*0"
  65 LET D$(6)="0*00*00*0"
  70 PRINT INK VAL "6";AT VAL "1
0",VAL "9";"ROLL THE DICE-"
  75 FLASH 5GN PI: BRIGHT NOT PI
: PRINT INK VAL "5";AT VAL "12",
VAL "12";"ENTER 2": FLASH NOT PI
: BRIGHT 5GN PI
  80 INPUT A
  90 CLS
 100 IF A=NOT PI THEN STOP
 105 IF A<>VAL "2" THEN GO TO 1e
3
 110 FOR X=5GN PI TO A
 120 LET B=INT (5GN PI+6*RND)
 130 FOR Y=5GN PI TO INT PI*INT
PI STEP INT PI
 140 PRINT D$(B,Y TO Y+VAL "2")
 150 NEXT Y: PRINT : NEXT X
 200 PRINT INK INT PI;AT VAL "20
",VAL "3";"Enter 0 to STOP"
 300 PLOT NOT PI,Y+140
 400 DRAW VAL "25",NOT PI: DRAW
NOT PI,VAL "25": DRAW VAL "-25",
NOT PI: DRAW NOT PI,VAL "-25"
 500 PLOT NOT PI,Y+109
 600 DRAW 25,0: DRAW 0,25: DRAW
-25,0: DRAW 0,-25
 505 GO TO VAL "70"
1000 CLS : FLASH 5GN PI: PRINT P
APER VAL "5": INK INT PI*INT PI;
AT VAL "10",VAL "9";"Try Again!!
!": FLASH NOT PI: PAUSE 60: CLS
: GO TO 70
5000 SAVE "DICE-II" LINE 3
```

courtesy of QuarterS/WMJ Data Systems

SIMPLE SNAP-GRID
for Paul Bingham's
ULTRA-EASY DESIGNER GRAPHICS

by Herb Bowers

When I first saw Paul Bingham's Ultra-Easy Designer Graphics program in the July/August '86 issue of TIME DESIGNS, I was not very impressed. I thought it was another one of those "ho-hum" UDG programs for the TS2068; difficult to type in and even more difficult to use. So I zapped right on by it, having given it only a token glance.

About a month later, I was in the midst of writing a program called Money Machine II and was attempting to design a simple animated graphic and found that none of the UDG utilities that I had would do the trick. I remembered having seen Paul's article and the words "Ultra-Easy" must have made its impression on me. If it's easy, I'm all for it.

I located my TDM under the pile of notes, abandoned LLISTing and other debris that seems to accumulate on my desk when I am hard at "work", and after an hour or so of hunt-n-pecking, I had Paul's program up and running. No bugs the first time through.

I had so much fun playing with the utility that I almost forgot my original purpose. I set to work on my animated graphic and then came a rude awakening. I was having a little difficulty in getting things lined up properly on the screen and had to do a lot of back-spacing, deleting and redarkening of pixels. A cursor travels in straight lines, but my eye-balls don't. Thats why we have rulers. A straight edge held against the screen helped, but was a little awkward.

I at once set out to design an optional grid. The one I came up with was optional but permanent because once you put it in, it was there to stay. I also devised a tinted area to seperate the four cells. It worked well and served my purpose for the time being...it could be put in Paul's program after some work was already in progress, and it didn't interfere with the CODE function of the program.

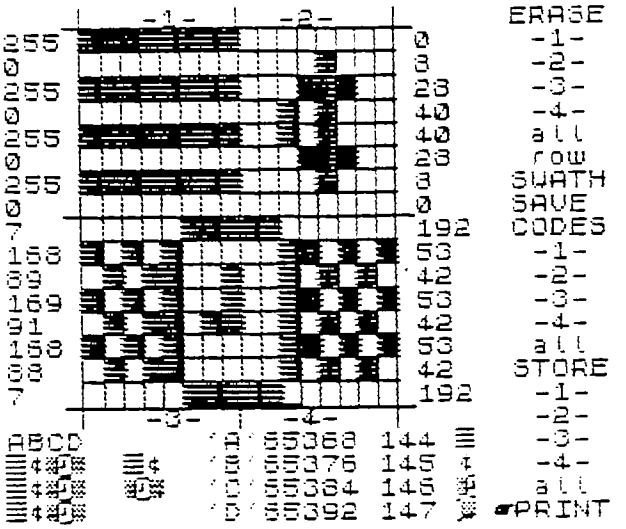
I sent a copy of the enchancement to Paul. He liked it, and admitted that he had fooled around with the grid idea but the little cross in the middle of the screen was all that was left of it. He told me that one thing he would want to add to his UDG program is what is called a "snap-grid". He went on to explain that in CAD programs, it is a simple grid that one can turn on and off at will. He said that the tricky part of doing this is avoiding "goofing up" work already on the screen. He asked me if I was up to it.

I have never seen a CAD program but I was determined that once I had MONEY MACHINE II in the can, that I was going to develop this "snap-grid".

The "snap-grid" is now completed and is presented here for you to add to Mr. Binghams already great program. You can type it directly into the program that was published in TDM or type it separately, SAVE it, then MERGE it with the program. Incidently, the enhancement may also be MERGEd with Paul's "bells-n-whistles" version which appeared in the November/December '86 issue of TDM, if you happen to have that.

Lines 26, 46, 400 and 8105 only require a slight addition to the original program lines. All other lines are new. Naturally, if you are going to MERGE the enhancement, than all lines must be copied in their entirety.

An actual SCREEN$ from the enhancement with the "snap-grid" in place is presented here. I have re-entered the UDG's that were shown in the original article. Mr. Bingham showed us a "venetian blind", a cent mark, and appropriately enough, a watch(Timex?)



created with a multi-CHR$ UDG. The very simple "venetian blind" configuration would really not require a grid to design, however, the grid is a great aid in designing the cent mark and the watch.

Take Paul's watch (please!)...with the grid in place, you can move the "phantom cursor" around each of the cells and darken the appropriate pixels without any back-tracking in less than half the time it takes to do it without the grid. The grid would also be a nifty device to use if you wanted to add some additional links to the watch band at a later time with additional multi-CHR$ UDGs. Above all, the grid is absolutely essential in designing simple animated graphics from BASIC.

The only additional documentation necessary for operation of the grid is that your KEY DESCRIPTION list should now contain a description for the "2" key which is very simply: "2 snap-grid toggle".

If you missed the July/August '86 or the November/December '86 issues of TIME DESIGNS which contained the original LLISTing (and enhanced version in the latter) for ULTRA-EASY DESIGNER GRAPHICS, you can order copies for just $3 each postage paid from the publisher.

Oh, just one other little thing. If you find that you have been losing the last byte of CHR$ 163, (UDG "T"), when you SAVE your UDGs...you can fix this by changing the number 159 located in lines 28 and 29 of the original published program, to 160 (in the enhanced version, the line numbers are 28 and 370).

If you have any questions, you can write to me at 2588 Woodshire Cir., Chesapeake, VA 23323.

```
0>REM

"ULTRA-EASY DESIGNER GRAPHICS"
  version 1.0 by P. bingham
   "SNAP-GRID" ENHANCEMENT
     by: herb bowers 10/86

  26 LET grid=0: PAPER 7: LET k=
5+(8*INT (px/13)): FOR h=k TO k+
7: PRINT AT py,h;CHR$ 128: NEXT
h: RETURN : REM **Add LET grid=0
        statement #1 to existing
        line**

  46 NEXT g: LET y1=y1+1: NEXT m
: NEXT t: LET grid=0: RETURN :
REM **add grid 0 statement #5 to
      existing line**

 103 LET grid=0
 195 IF CODE INKEY$=50 THEN GO S
UB 9300
 315 IF a(t)=5 THEN LET a(t)=7
```

```
 400 PAPER 7: FOR h=x TO x+7: FO
R t=y TO y+7: PRINT AT h,t;CHR$
128: NEXT t: NEXT h: LET grid=0:
 RETURN :          REM **Add grid=0
        statement #8 to existing
        line**

8102 LET py1=168-py*8: LET px1=p
x*8

8105 IF at=7 OR at=5 THEN PRINT
PAPER at;AT py,px;CHR$ 134: PRIN
T AT py,px;CHR$ 137: PRINT AT py
,px;CHR$ 128: REM **Line change
          only to add OR at=5 in
          statement #1**

8107 IF grid THEN PLOT px1,py1:
DRAW 8,0: PLOT px1,py1: DRAW 0,8

9135 LET grid=0
9300 IF grid THEN GO TO 9335
9305 FOR t=48 TO 160 STEP 8: PLO
T t,40: DRAW 0,127: NEXT t
9310 FOR t=48 TO 160 STEP 8: PLO
T 40,t: DRAW 127,0: NEXT t
9315 FOR h=12 TO 13: FOR t=1 TO
16: IF ((ATTR (t,h)/8)<>0) THEN
PRINT PAPER 5; OVER 1;AT t,h;" "
9320 NEXT t: NEXT h: LET grid=1
9325 FOR h=8 TO 9: FOR t=5 TO 20
: IF ((ATTR (h,t)/8)<>0) THEN PR
INT PAPER 5; OVER 1;AT h,t;" "
9330 NEXT t: NEXT h: RETURN
9335 FOR h=5 TO 20: FOR t=1 TO 1
6: IF ((ATTR (t,h)/8)<>0) THEN P
RINT PAPER 7;AT t,h;" "
9340 NEXT t: NEXT h
9345 LET grid=0: RETURN
```

DRAW POKER

BY

WILLIAM C. ANDREWS

This TS2068 program is a FIVE CARD DRAW POKER game, playing against the computer as the dealer. After making a bet you are dealt five cards. You may discard or hold any number of these cards. Discards will be replaced and the dealer will determine a winner. You must have a pair of JACKS or better to win. The payoff depends on the the value of your win. The computer is also your banker to watch your finances.

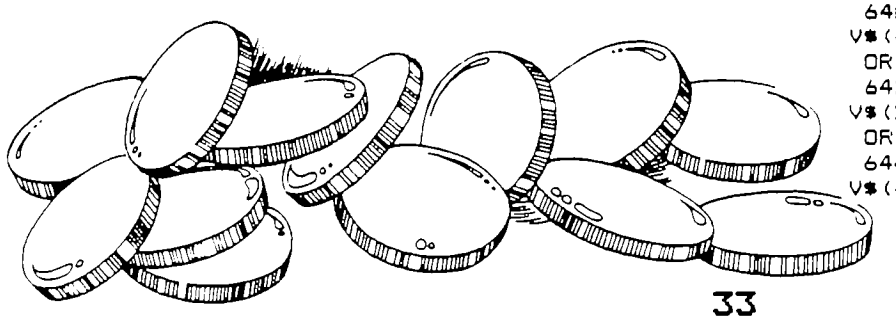
When typing the program, note that letters in quotes in lines 140, 160, 180, 7020, 9991 and the last letter in lines 8001 to 8052 are in GRAPHIC mode for UDG's. To clear screen of color to work on the program, use GOTO 9999.

A copy of this program on tape is available from me, for $10.00 postage paid if you do not wish to type in the listing: 30 Oak Knoll Drive, San Anselmo, CA 94960.

```
  5 CLS : LET S=100
 10 GO SUB 9000
 20 GO SUB 8000
 30 BORDER 4: PAPER 4: CLS
 35 GO SUB 1200
 40 PRINT AT 9,13; INK 1;"▀▀▀▀
▀";AT 10,13; PAPER 2; INK 9; FLA
SH 1;" DEAL ";AT 11,13; PAPER 8;
 INK 1; FLASH 0;"▀▀▀▀▀▀▀": PAUSE
200
 50 FOR N=9 TO 11: PRINT AT N,1
3;"       ": NEXT N
 60 DIM V$(52,2)
 70 DIM D$(52,2)
 80 DIM X$(5,1)
 90 LET P=0
100 LET Z=1
110 RANDOMIZE
120 FOR A=2 TO 30 STEP 6
130 FOR B=1 TO 5
140 PRINT AT 8,A; PAPER 7;"FJJJ
G"
150 FOR N=9 TO 13
160 PRINT AT N,A; PAPER 7;"K
K"
170 NEXT N
180 PRINT AT 14,A; PAPER 7;"HJJ
JI"
190 NEXT B
195 BEEP .01,15
200 GO SUB 7000
```

```
210 PRINT  PAPER 7;AT 9,A+1; IN
K X;A$(F,4);AT 11,A+1; INK 0;A$(
F,2 TO 3);AT 13,A+3; INK X;A$(
F,4)
215 GO SUB 7030
220 LET A$(F,1)="1"
230 LET D$(N)=A$(F,2 TO 3)
240 LET V$(Z)=D$(N)
250 LET X$(Z)=A$(F,4)
260 LET Z=Z+1
270 NEXT A
280 LET P=0: GO SUB 380
290 FOR N=2 TO 30 STEP 6
300 PRINT AT 15,N; PAPER 7; FLA
SH 1;"HOLD?";AT 20,14; PAPER 7;"
(Y/N)": PAUSE 0: FLASH 0: IF INK
EY$="Y" OR INKEY$="y" THEN  GO S
UB 380
310 IF INKEY$<>"Y" AND INKEY$<>
"y" THEN  GO SUB 342
330 IF INKEY$<>"Y" AND INKEY$<>
"y" THEN  LET D$(N)="  "
340 NEXT N
341 GO TO 350
342 FOR I=8 TO 15
343 PRINT AT I,N;"      "
344 NEXT I
345 LET D$(N)="  "
346 RETURN
350 PRINT AT 17,0;"
                    ";AT 20,14;"
    "
360 PRINT AT 16,13; INK 1;"▀▀▀▀
▀";AT 17,13; PAPER 2; INK 9; FL
ASH 1;" DEAL ";AT 18,13; PAPER 8
; INK 1; FLASH 0;"▀▀▀▀▀▀▀": PAUSE
 0
370 PRINT AT 16,13;"       ";AT
17,13;"       ";AT 18,13;"       "
; GO TO 390
380 PRINT AT 15,N; PAPER 1; INK
9;"HOLD ": RETURN
390 LET Z=0
400 PRINT AT 20,0;"
                    "
405 FOR N=2 TO 30 STEP 6
410 PRINT AT 8,N; PAPER 7;"FJJJ
G"
```

```
 411  FOR K=9 TO 13
 412 PRINT  PAPER 7;AT K,N;"K";A
T K,N+4;"K"
 413 IF SCREEN$ (15,N)=" " THEN
 PRINT AT K,N+1; PAPER 7;"    "
 414 NEXT K
 415 PRINT AT 14,N; PAPER 7;"HJJ
JI"
 420 LET Z=Z+1
 430 IF SCREEN$ (15,N)="H" THEN
 LET D$(Z)=V$(N)
 440 IF SCREEN$ (15,N)="H" THEN
 GO TO 520
 450 GO SUB 7000
 470 IF SCREEN$ (15,N)=" " THEN
 PRINT  PAPER 7;AT 9,N+1; INK X;
A$(F,4);AT 11,N+1; INK 0;A$(F,2
TO 3);AT 13,N+3; INK X;A$(F,4)
: BEEP .01,15
 475 GO SUB 7030
 480 LET A$(F,1)="1"
 490 LET D$(N)=A$(F,2 TO 3)
 500 LET V$(Z)=D$(N)
 510 LET X$(Z)=A$(F,4)
 520 NEXT N
 525 PRINT AT 15,0;"
                    "
 530 LET P=0: GO SUB 580
 535 IF P>=1 AND VAL J$>10 OR VA
L J$=1 THEN  GO SUB 900
 536 IF P>=2 THEN  GO SUB 900
 540 IF P=0 OR P<=1 AND VAL J$<1
1 AND VAL J$<>1 THEN  PRINT AT 1
9,7; PAPER 1; INK 9;" SORRY---TR
Y AGAIN? ";AT 4,20;"LOST": BEEP
.5,-24: BEEP 1,-30
 550 IF P<=1 AND VAL J$<11 AND V
AL J$>=1 THEN  PRINT AT 6,0; PAP
ER 5; INK 0; FLASH 1;" PAIR OF J
ACKS OR BETTER TO WIN "
 560 PAUSE 0
 570 CLS : PRINT AT 10,10; PAPER
 2; INK 6; FLASH 1;" SHUFFLING "
: GO TO 20
 580 PRINT AT 17,0;"
                    "
 590 IF P=0 THEN  IF V$(1)=V$(2)
 OR V$(1)=V$(3) OR V$(1)=V$(4) O
R V$(1)=V$(5) THEN  LET P=1: LET
 J$=V$(1)
 592 IF P=0 THEN  IF V$(2)=V$(3)
 OR V$(2)=V$(4) OR V$(2)=V$(5) T
HEN  LET P=1: LET J$=V$(2)
 594 IF P=0 THEN  IF V$(3)=V$(4)
 OR V$(3)=V$(5) THEN  LET P=1: L
ET J$=V$(3)
 596 IF P=0 THEN  IF V$(4)=V$(5)
 THEN  LET P=1: LET J$=V$(4)
 598 IF P=1 AND V$(1)=V$(2) AND
V$(3)<>V$(1) AND V$(3)<>V$(2) TH
EN  IF V$(3)=V$(4) OR V$(3)=V$(5
) OR V$(4)=V$(5) THEN  LET P=2
 600 IF P=1 AND V$(1)=V$(3) AND
V$(4)<>V$(1) AND V$(4)<>V$(3) TH
EN  IF V$(4)=V$(2) OR V$(4)=V$(5
) OR V$(2)=V$(5) THEN  LET P=2
 602 IF P=1 AND V$(1)=V$(4) AND
V$(5)<>V$(1) AND V$(5)<>V$(4) TH
EN  IF V$(5)=V$(2) OR V$(5)=V$(3
) OR V$(3)=V$(5) THEN  LET P=2
 604 IF P=1 AND V$(1)=V$(5) AND
V$(2)<>V$(1) AND V$(2)<>V$(5) TH
EN  IF V$(2)=V$(3) OR V$(2)=V$(4
) OR V$(3)=V$(4) THEN  LET P=2
 606 IF P=1 AND V$(2)=V$(3) AND
V$(1)<>V$(2) AND V$(1)<>V$(3) TH
EN  IF V$(1)=V$(4) OR V$(1)=V$(5
) OR V$(4)=V$(5) THEN  LET P=2
 608 IF P=1 AND V$(2)=V$(4) AND
V$(1)<>V$(2) AND V$(1)<>V$(4) TH
EN  IF V$(1)=V$(3) OR V$(1)=V$(5
) OR V$(3)=V$(5) THEN  LET P=2
 610 IF P=1 AND V$(2)=V$(5) AND
V$(1)<>V$(2) AND V$(1)<>V$(5) TH
EN  IF V$(1)=V$(3) OR V$(1)=V$(4
) OR V$(3)=V$(4) THEN  LET P=2
 612 IF P=1 AND V$(3)=V$(4) AND
V$(1)<>V$(3) AND V$(1)<>V$(4) TH
EN  IF V$(1)=V$(2) OR V$(1)=V$(5
) OR V$(2)=V$(5) THEN  LET P=2
 614 IF P=1 AND V$(3)=V$(5) AND
V$(1)<>V$(3) AND V$(1)<>V$(5) TH
EN  IF V$(1)=V$(2) OR V$(1)=V$(4
) OR V$(2)=V$(4) THEN  LET P=2
 616 IF P=1 AND V$(4)=V$(5) AND
V$(1)<>V$(4) AND V$(1)<>V$(5) TH
EN  IF V$(1)=V$(2) OR V$(1)=V$(5
) OR V$(2)=V$(3) THEN  LET P=2
 618 IF P=1 AND V$(1)=V$(2) THEN
   IF V$(3)=V$(1) OR V$(4)=V$(1)
OR V$(5)=V$(1) THEN  LET P=3
 620 IF P=1 AND V$(1)=V$(3) THEN
   IF V$(2)=V$(1) OR V$(4)=V$(1)
OR V$(5)=V$(1) THEN  LET P=3
 622 IF P=1 AND V$(1)=V$(4) THEN
   IF V$(2)=V$(1) OR V$(3)=V$(1)
OR V$(5)=V$(1) THEN  LET P=3
 624 IF P=1 AND V$(1)=V$(5) THEN
   IF V$(2)=V$(1) OR V$(3)=V$(1)
OR V$(4)=V$(1) THEN  LET P=3
 626 IF P=1 AND V$(2)=V$(3) THEN
   IF V$(1)=V$(2) OR V$(4)=V$(2)
OR V$(5)=V$(2) THEN  LET P=3
 628 IF P=1 AND V$(2)=V$(4) THEN
   IF V$(1)=V$(2) OR V$(3)=V$(2)
OR V$(5)=V$(2) THEN  LET P=3
 630 IF P=1 AND V$(3)=V$(4) THEN
   IF V$(1)=V$(3) OR V$(2)=V$(3)
OR V$(5)=V$(3) THEN  LET P=3
 632 IF P=1 AND V$(3)=V$(5) THEN
   IF V$(1)=V$(3) OR V$(2)=V$(3)
OR V$(4)=V$(3) THEN  LET P=3
 634 IF P=1 AND V$(4)=V$(5) THEN
   IF V$(1)=V$(4) OR V$(2)=V$(4)
OR V$(3)=V$(4) THEN  LET P=3
 636 IF P=2 AND V$(1)=V$(2) AND
V$(3)=V$(4) THEN  IF V$(5)=V$(1)
 OR V$(5)=V$(3) THEN  LET P=5
 638 IF P=2 AND V$(1)=V$(2) AND
V$(3)=V$(5) THEN  IF V$(4)=V$(1)
 OR V$(4)=V$(3) THEN  LET P=5
 640 IF P=2 AND V$(1)=V$(2) AND
V$(4)=V$(5) THEN  IF V$(3)=V$(1)
 OR V$(3)=V$(4) THEN  LET P=5
 642 IF P=2 AND V$(1)=V$(3) AND
V$(2)=V$(5) THEN  IF V$(4)=V$(1)
 OR V$(4)=V$(5) THEN  LET P=5
 644 IF P=2 AND V$(1)=V$(3) AND
V$(4)=V$(5) THEN  IF V$(2)=V$(1)
```



```
 OR V$(2)=V$(4) THEN  LET P=5
 646 IF P=2 AND V$(1)=V$(4) AND
V$(2)=V$(3) THEN  IF V$(5)=V$(1)
 OR V$(5)=V$(2) THEN  LET P=5
 648 IF P=2 AND V$(1)=V$(4) AND
V$(2)=V$(5) THEN  IF V$(3)=V$(1)
 OR V$(3)=V$(2) THEN  LET P=5
 650 IF P=2 AND V$(1)=V$(5) AND
V$(2)=V$(3) THEN  IF V$(4)=V$(1)
 OR V$(4)=V$(2) THEN  LET P=5
 652 IF P=2 AND V$(1)=V$(5) AND
V$(2)=V$(4) THEN  IF V$(3)=V$(1)
 OR V$(3)=V$(2) THEN  LET P=5
 654 IF P=2 AND V$(1)=V$(5) AND
V$(3)=V$(4) THEN  IF V$(2)=V$(1)
 OR V$(2)=V$(3) THEN  LET P=5
 656 IF P=2 AND V$(2)=V$(3) AND
V$(4)=V$(5) THEN  IF V$(1)=V$(2)
 OR V$(1)=V$(4) THEN  LET P=5
 658 IF P=2 AND V$(2)=V$(4) AND
V$(3)=V$(5) THEN  IF V$(1)=V$(2)
 OR V$(1)=V$(3) THEN  LET P=5
 660 IF P=2 AND V$(2)=V$(5) AND
V$(3)=V$(4) THEN  IF V$(1)=V$(2)
 OR V$(1)=V$(3) THEN  LET P=5
 662 IF P=3 AND V$(1)=V$(2) AND
V$(1)=V$(3) THEN  IF V$(4)=V$(1)
 OR V$(5)=V$(1) THEN  LET P=4
 664 IF P=3 AND V$(1)=V$(2) AND
V$(1)=V$(4) THEN  IF V$(3)=V$(1)
 OR V$(5)=V$(1) THEN  LET P=4
 666 IF P=3 AND V$(1)=V$(3) AND
V$(1)=V$(4) THEN  IF V$(2)=V$(1)
 OR V$(5)=V$(1) THEN  LET P=4
 668 IF P=3 AND V$(1)=V$(2) AND
V$(1)=V$(5) THEN  IF V$(3)=V$(1)
 OR V$(4)=V$(1) THEN  LET P=4
 670 IF P=3 AND V$(1)=V$(3) AND
V$(1)=V$(5) THEN  IF V$(2)=V$(1)
 OR V$(4)=V$(1) THEN  LET P=4
 672 IF P=3 AND V$(1)=V$(4) AND
V$(1)=V$(5) THEN  IF V$(2)=V$(1)
 OR V$(3)=V$(1) THEN  LET P=4
 674 IF P=3 AND V$(2)=V$(3) AND
V$(2)=V$(4) THEN  IF V$(1)=V$(2)
 OR V$(5)=V$(2) THEN  LET P=4
 676 IF P=3 AND V$(2)=V$(4) AND
V$(2)=V$(5) THEN  IF V$(1)=V$(2)
 OR V$(3)=V$(2) THEN  LET P=4
 678 IF P=3 AND V$(3)=V$(4) AND
V$(3)=V$(5) THEN  IF V$(1)=V$(3)
 OR V$(2)=V$(3) THEN  LET P=4
 680 IF P=3 AND V$(1)=V$(2) AND
V$(1)=V$(3) THEN  IF V$(4)=V$(5)
 THEN  LET P=5
 682 IF P=3 AND V$(1)=V$(2) AND
V$(1)=V$(4) THEN  IF V$(3)=V$(5)
 THEN  LET P=5
 684 IF P=3 AND V$(1)=V$(2) AND
V$(1)=V$(5) THEN  IF V$(3)=V$(4)
 THEN  LET P=5
 686 IF P=3 AND V$(1)=V$(3) AND
V$(1)=V$(4) THEN  IF V$(2)=V$(5)
 THEN  LET P=5
 688 IF P=3 AND V$(1)=V$(3) AND
V$(1)=V$(5) THEN  IF V$(2)=V$(4)
 THEN  LET P=5
 690 IF P=3 AND V$(1)=V$(4) AND
V$(1)=V$(5) THEN  IF V$(2)=V$(3)
 THEN  LET P=5
 692 IF P=3 AND V$(2)=V$(3) AND
V$(2)=V$(4) THEN  IF V$(1)=V$(5)
 THEN  LET P=5
 694 IF P=3 AND V$(2)=V$(3) AND
V$(2)=V$(5) THEN  IF V$(1)=V$(4)
 THEN  LET P=5
 696 IF P=3 AND V$(2)=V$(4) AND
V$(2)=V$(5) THEN  IF V$(1)=V$(3)
 THEN  LET P=5
 698 IF P=3 AND V$(3)=V$(4) AND
V$(3)=V$(5) THEN  IF V$(1)=V$(2)
 THEN  LET P=5
 780 IF X$(1)=X$(2) AND X$(2)=X$
(3) AND X$(3)=X$(4) AND X$(4)=X$
(5) THEN  LET P=6
 785 GO SUB 1000
 790 IF P=1 THEN  PRINT AT 17,11
; PAPER 2; INK 9;" ONE PAIR ": B
EEP .5,10
 800 IF P=2 THEN  PRINT AT 17,11
; PAPER 2; INK 9;" TWO PAIR ": B
EEP .5,10: BEEP .5,10
 810 IF P=3 THEN  PRINT AT 17,8;
 PAPER 2; INK 9;" THREE OF A KIN
D ": BEEP .5,10: BEEP .5,10: BEE
P .5,10
 820 IF P=4 THEN  PRINT AT 17,8;
 PAPER 2; INK 9;" FOUR OF A KIND
 ": BEEP .5,10: BEEP .5,10: BEEP
 .5,10: BEEP .5,10
 830 IF P=5 THEN  PRINT AT 17,10
; PAPER 2; INK 9;" FULL HOUSE ":
 BEEP .3,10: BEEP .3,10: BEEP .3
,10: BEEP .3,10: BEEP .3,10
 840 IF P=6 THEN  PRINT AT 17,13
; PAPER 2; INK 9;" FLUSH ": FOR
N=1 TO 20: BEEP .01,N: NEXT N
 845 IF P=7 THEN  PRINT AT 17,11
; PAPER 2; INK 9;" STRAIGHT ": F
OR N=1 TO 20: BEEP .01,N: NEXT N
 850 RETURN
 900 PRINT AT 19,13;" PAYS ";AT
21,13;P;" TO 1"
 910 LET Q=W*P: PRINT AT 4,20; P
APER 1; INK 9;"WON--"; PAPER 6;"
$ ";Q;" ": LET S=S+Q+W
 920 PAUSE 60: PRINT AT 2,14;"
  ";AT 2,14; PAPER 6; INK 9;S;"
": RETURN
1000 DIM A(5): FOR N=1 TO 5
1005 LET A(N)=VAL V$(N)
1010 NEXT N: GO SUB 1030: RETURN

1015 DIM A(5): FOR N=1 TO 5
1020 LET A(N)=D(N)
1025 NEXT N: GO SUB 1030: RETURN

1030 LET STR=0: FOR N=1 TO 4
1035 FOR M=1 TO 5-N
1040 LET C=A(M)
1045 LET D=A(M+1)
1050 IF C<=D THEN  GO TO 1065
1055 LET A(M)=D
1060 LET A(M+1)=C
1065 NEXT M
1070 NEXT N
1075 FOR N=2 TO 5
1080 IF A(N)=A(N-1)+1 THEN  GO T
O 1090
1085 RETURN
1090 NEXT N
1095 LET STR=1: LET P=7: RETURN
1200 PRINT AT 2,1; PAPER 1; INK
9;" YOU HAVE- "; PAPER 6; INK 9;
"$ ";S;" ": PAUSE 60
1205 IF S<=0 THEN  PRINT AT 10,0
;" YOU'RE BROKE---SEE YOUR BANKE
R ";AT 12,8;" COME BACK AGAIN. "
: PAUSE 0: IF S<=0 THEN  STOP
1210 PRINT AT 4,15; PAPER 1; INK
 9;" YOU BET--"; PAPER 6; INK 1;
 FLASH 1;"$ ? "
1220 INPUT  PAPER 6; INK 9;" LIM
IT= $ 10   ";W
1222 IF W<1 THEN  GO TO 1220
1225 IF W>10 THEN  GO TO 1220
```

```
1230 PRINT AT 4,24; PAPER 6; INK
 9;"$ ";W;"   "
1232 LET S=S-W
1235 PRINT AT 6,0; PAPER 2; INK
9; BRIGHT 1;" PAIR OF JACKS OR B
ETTER TO WIN "
1240 PRINT AT 2,14;"              "
;AT 2,14; PAPER 6; INK 9;S;" "
1250 RETURN
7000 LET F=INT (RND*52)+1
7010 IF A$(F,1)<>"0" THEN  GO TO
 7000
7020 LET X=0: IF A$(F,4)="A" OR
A$(F,4)="D" THEN  LET X=2
7025 RETURN
7030 IF A$(F,2 TO 3)=" A" THEN
LET A$(F,2 TO 3)="14"
7040 IF A$(F,2 TO 3)=" J" THEN
LET A$(F,2 TO 3)="11"
7050 IF A$(F,2 TO 3)=" Q" THEN
LET A$(F,2 TO 3)="12"
7055 IF A$(F,2 TO 3)=" K" THEN
LET A$(F,2 TO 3)="13"
7060 IF A$(F,2)=" " THEN  LET A$
(F,2)="0"
7070 RETURN
8000 DIM A$(52,4): DIM J$(2): LE
T J$="00"
8001 LET A$(1)="0 AA"
8002 LET A$(2)="0 2A"
8003 LET A$(3)="0 3A"
8004 LET A$(4)="0 4A"
8005 LET A$(5)="0 5A"
8006 LET A$(6)="0 6A"
8007 LET A$(7)="0 7A"
8008 LET A$(8)="0 8A"
8009 LET A$(9)="0 9A"
8010 LET A$(10)="010A"
8011 LET A$(11)="0 JA"
8012 LET A$(12)="0 QA"
8013 LET A$(13)="0 KA"
8014 LET A$(14)="0 AB"
8015 LET A$(15)="0 2B"
8016 LET A$(16)="0 3B"
8017 LET A$(17)="0 4B"
8018 LET A$(18)="0 5B"
8019 LET A$(19)="0 6B"
8020 LET A$(20)="0 7B"
8021 LET A$(21)="0 8B"
8022 LET A$(22)="0 9B"
8023 LET A$(23)="010B"
8024 LET A$(24)="0 JB"
8025 LET A$(25)="0 QB"
8026 LET A$(26)="0 KB"
8027 LET A$(27)="0 AC"
8028 LET A$(28)="0 2C"
8029 LET A$(29)="0 3C"
8030 LET A$(30)="0 4C"
8031 LET A$(31)="0 5C"
8032 LET A$(32)="0 6C"
8033 LET A$(33)="0 7C"
8034 LET A$(34)="0 8C"
8035 LET A$(35)="0 9C"
8036 LET A$(36)="010C"
8037 LET A$(37)="0 JC"
8038 LET A$(38)="0 QC"
8039 LET A$(39)="0 KC"
8040 LET A$(40)="0 AD"
8041 LET A$(41)="0 2D"
8042 LET A$(42)="0 3D"
8043 LET A$(43)="0 4D"
8044 LET A$(44)="0 5D"
8045 LET A$(45)="0 6D"
8046 LET A$(46)="0 7D"
8047 LET A$(47)="0 8D"
8048 LET A$(48)="0 9D"
8049 LET A$(49)="010D"
8050 LET A$(50)="0 JD"
8051 LET A$(51)="0 QD"
8052 LET A$(52)="0 KD"
8053 RETURN
9000 RESTORE : FOR a=USR "a" TO
USR "k"+7
9010 READ h: POKE a,h
9020 NEXT a: RETURN
9030 DATA 24,60,126,255,255,126,
60,24
9040 DATA 28,28,8,107,127,107,8,
28
9050 DATA 16,56,124,254,254,254,
16,56
9060 DATA 102,255,255,255,255,12
6,60,24
9070 DATA 204,204,51,51,204,204,
51,51
9080 DATA 0,0,0,7,15,12,24,24
9090 DATA 0,0,0,192,240,48,24,24
9100 DATA 24,24,12,15,7,0,0,0
9110 DATA 24,24,48,240,192,0,0,0
9120 DATA 0,0,0,255,255,0,0,0
9130 DATA 24,24,24,24,24,24,24,2
4
9990 REM a b c d e f g h i j k
9991 REM A B C D E F G H I J K
9992 BORDER 7: PAPER 7: INK 0: C
LS : STOP
9994 CLS : SAVE "DRAW POKER" LIN
E 0
9995 GO TO 0
9999 BORDER 7: PAPER 7: INK 0: B
RIGHT 0: CLS : LIST
```

BLACK JACK

BY

WILLIAM C. ANDREWS

This is a TS2068 version of Las Vegas BLACK JACK except for 'doubling' or 'splitting'. It will offer an insurance bet against dealer's BLACK JACK. The computer is the dealer and will monitor the game's progress. The program is self-starting, no dealt card will be duplicated, and a shuffle will occur for each new game.

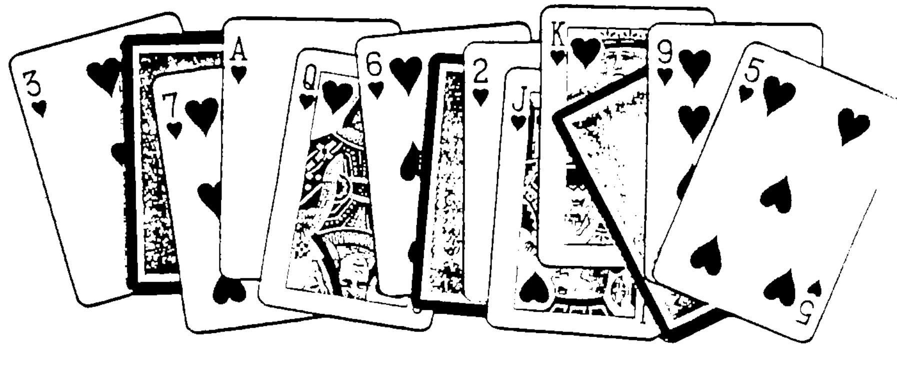
Letters in quotes in lines 1800, 1820, 1840, 1930, and 2000 are in GRAPHIC mode for UDG's. This is shown in line 2770 which are also in graphic mode. The suit of each card is the last letter (graphic mode) in lines 2090 to 2600. A GOTO 2780 will clear all color from the screen to better study the lines.

Tired programmers can get a taped copy of the program from me for $7.50 postage paid, at: 30 Oak Knoll Drive, San Anselmo, CA 94960.

```
  10 LET ST=200
  20 RANDOMIZE
  30 BORDER 4: PAPER 7: CLS
  40 GO SUB 2080
  50 IF ST=0 THEN  GO TO 1340
  60 PRINT AT 0,0; PAPER 4;"
                            ";AT
 0,10; PAPER 2; INK 9; BRIGHT 1;
" BLACKJACK "
  70 INK 1
  80 LET K$="▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄
▄▄▄▄▄▄▄▄▄▄"
  90 LET L$="▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀
▀▀▀▀▀▀▀▀▀▀▀"
 100 PRINT  PAPER 4;K$
 110 PRINT "▌  YOUR HAND       DE
ALERS HAND  ▐"
```

```
 120 PRINT AT 3,0;L$;AT 3,0;"▛";
AT 3,31;"▜"
 130 FOR I=1 TO 6
 140 INK 1: PRINT "▌";TAB 31;"▐
";TAB 31;"▐"
 150 NEXT I
 160 FOR I=2 TO 14
 170 PRINT AT I,15;"▀"
 180 NEXT I
 190 PRINT L$
 200 PRINT AT 15,0;"▛";AT 15,31;
"▜"
 210 PRINT AT 16,0;"▌";AT 16,31;
"▐"
 220 PRINT AT 17,0;L$
 230 PRINT AT 15,12;"  BET
"
 240 FOR N=16 TO 18
 250 PRINT AT N,12;"▌       ▐"
 260 NEXT N
 270 PRINT AT 19,12;"▀▀▀▀▀▀▀▀"
 280 INK 0
 290 PRINT    INK 0;AT 18,0;"YOUR
 STAKE";AT 20,2;"$ ";ST
 300 PRINT  FLASH 1; INK 2;;AT 2
1,8;" MAKE YOUR BET "
 310 INPUT W
 320 IF W<1 THEN  GO TO 310
 330 IF W>ST THEN  PRINT  INK 0;
AT 16,13;"THATS";AT 17,13;" TOO
";AT 18,13;" MUCH"
 340 IF W>100 THEN  PRINT  INK 0
;AT 16,13;"SORRY";AT 17,13;"LIMI
T";AT 18,13;"$ 100": GO TO 310
 350 IF W>ST THEN  GO TO 310
 360 PRINT AT 21,8;"
    "
 370 PRINT AT 16,13;"      ";AT 1
7,13;"      ";AT 18,13;"      "; I
NK 0;AT 17,13;"$ ";W
 380 LET ST=ST-W
 390 PRINT AT 20,4;ST;"  "
 400 LET P1=0: LET P2=0: LET P3=
0: LET P4=0: LET P5=0
 410 LET D1=0: LET D2=0: LET D3=
0: LET D4=0: LET D5=0
 420 LET H=0: LET A=2: LET B=4:
GO SUB 1780
 430 LET P1=V
 440 LET A=8: GO SUB 1780
 450 LET P2=V
 460 LET P=P1+P2: IF P>21 AND P1
=11 OR P>21 AND P2=11 THEN  GO S
UB 720
 470 IF P=21 THEN  GO TO 1950
 480 LET H=1: LET A=18: GO SUB 1
780
 490 LET H=0: LET A=24: GO SUB 1
780
 500 LET A=2: LET D2=V
 510 GO SUB 1980: LET G=F: LET D
1=V: LET D=D1+D2: IF D>21 AND D1
=11 OR D>21 AND D2=11 THEN  GO S
UB 990
 520 IF D=21 THEN  LET A=18: GO
SUB 1790: GO TO 1950
 530 LET IN=0
 540 IF D2=11 THEN  GO SUB 1530
 550 GO TO 850
 560 PRINT  FLASH 1; INK 2;AT 21
,9;" HIT? (Y/N) "
 570 PAUSE 0
 580 IF INKEY$="Y" THEN  LET B=8
: GO TO 650
 590 IF INKEY$<>"N" THEN  GO TO
570
 600 PRINT  INK 0;AT 16,1;"STAY
ON ";P;" ";AT 21,9;"
"
 610 LET B=4: LET A=18: LET F=G:
 GO SUB 1790:
 620 IF IN>0 AND V<>10 THEN  GO
SUB 1710
 630 IF P>21 THEN  GO TO 1260
 640 GO TO 790
 650 IF A=3 THEN  LET A=5
 660 IF A=6 THEN  LET A=8
 670 IF A=9 THEN  LET A=11
 680 LET B=8: GO SUB 1780
 690 IF A=3 THEN  LET P3=V: LET
P=P+P3: GO TO 720
 700 IF A=6 THEN  LET P4=V: LET
P=P+P4: GO TO 720
 710 IF A=9 THEN  LET P5=V: LET
P=P+P5: GO TO 720
 720 IF P>21 AND P1=11 THEN  LET
 P1=1
 730 IF P>21 AND P2=11 THEN  LET
 P2=1
 740 IF P>21 AND P3=11 THEN  LET
 P3=1
 750 IF P>21 AND P4=11 THEN  LET
 P4=1
 760 IF P>21 AND P5=11 THEN  LET
 P5=1
 770 LET P=P1+P2+P3+P4+P5
 780 GO TO 850
 790 GO SUB 1040
 800 GO SUB 900: LET B=8
 810 IF A=19 THEN  LET A=21
 820 IF A=22 THEN  LET A=24
 830 IF A=25 THEN  LET A=27
 840 GO SUB 1780: GO TO 900
 850 PRINT AT 16,1;"YOU HAVE ";P
 860 IF B=4 THEN  PRINT AT 16,20
;"I SHOW ";D2
 870 IF P=21 AND B=4 THEN  GO TO
 1950
 880 IF P>21 THEN  GO TO 1130
 890 GO TO 560
 900 LET B=8: LET A=18: GO SUB 1
780
 910 LET D3=V: LET D=D+D3: IF D>
21 THEN  GO SUB 990
 920 GO SUB 1040: IF A=19 THEN
LET A=21: GO SUB 1780
 930 LET D4=V: LET D=D+D4: IF D>
21 THEN  GO SUB 990
 940 GO SUB 1040: IF A=22 THEN
LET A=24: GO SUB 1780
 950 LET D5=V: LET D=D+D5: IF D>
21 THEN  GO SUB 990
 960 GO SUB 1040: IF A=25 THEN
LET A=27: GO SUB 1780
 970 GO TO 1040
 980 IF IN>0 THEN  GO SUB 1710
 990 IF D1=11 THEN  LET D1=1
1000 IF D2=11 THEN  LET D2=1
1010 IF D3=11 THEN  LET D3=1
1020 IF D4=11 THEN  LET D4=1
1030 IF D5=11 THEN  LET D5=1
1040 LET D=0: LET D=D1+D2+D3+D4+
D5: PRINT AT 16,20;"I HAVE ";D:
PAUSE 100
1050 IF D=21 AND B=4 THEN  GO TO
 1950
1060 IF D>=17 THEN  GO TO 1080
1070 RETURN
1080 IF D>21 THEN  GO TO 1170
1090 IF D>=17 AND D<P THEN  GO T
O 1170
1100 IF D>=17 AND P>D THEN  GO T
O 1170
1110 IF D>P THEN  GO TO 1130
1120 IF D=P THEN  GO SUB 1220: P
RINT AT 17,14;"TIE": BEEP 1,
```

```
-12: PAUSE 100: LET ST=ST+W: GO
TO 1200
1130 GO SUB 1220
1140 PRINT AT 16,15;"I";AT
 18,14;"WIN": BEEP .3,10
1150 IF IN>0 THEN  GO TO 610
1160 GO TO 1260
1170 GO SUB 1220
1180 PRINT AT 16,14;"YOU
";AT 18,14;"WIN": BEEP .5,20
1190 LET ST=ST+2*W
1200 PRINT AT 20,4;ST
1210 GO TO 1260
1220 FOR N=15 TO 19
1230 PRINT AT N,12;"████████"
1240 NEXT N
1250 RETURN
1260 PRINT AT 19,10;"  ▀▀▀▀▀▀▀▀▀▀
";AT 20,10;"              ";AT 21,9
; PAPER 1; INK 9; FLASH 1;"  TRY
 AGAIN? "
1270 PAUSE 0
1280 IF INKEY$="N" THEN  GO TO 1
300
1290 GO TO 30
1300 PAPER 4: CLS
1310 PRINT AT 10,0; PAPER 2; INK
 9; BRIGHT 1;" OKAY--COME BACK A
GAIN SOMETIME "
1320 PAUSE 0
1330 STOP
1340 BORDER 5: PAPER 5: CLS
1350 PRINT AT 10,8; PAPER 2; INK
 9; BRIGHT 1;" YOU'RE BROKE--";A
T 12,5;" COME BACK SOMETIME "
1360 PAUSE 0
1370 STOP
1380 BORDER 3: PAPER 6: CLS :
1390 PRINT AT 0,0; INK 2;"▛▀▀▀▀▀▀▀▀
▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▜"
1400 FOR N=1 TO 20: PRINT AT N,3
1; INK 2;"█": NEXT N
1410 PRINT AT 21,0; INK 2:"▙▄▄▄▄▄▄
▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▟"
1420 PRINT  PAPER 5;AT 1,10:"▛▀▀▀
▀▀▀▀▀▀▀▜";AT 2,10:"▌BLACKJACK▐";
AT 3,10;"▙▄▄▄▄▄▄▄▄▄▟"
1430 PRINT AT 5,1:"THIS GAME OF
BLACKJACK FOLLOWS";AT 7,8;" LAS
VEGAS RULES." .
1440 PRINT AT 9,10:"LIMIT = $100
."
1450 PRINT AT 11,4;"DEALER WIL
L HOLD ON ""17""."
1460 PRINT AT 13,4;"""BLACKJACK"
" PAYS 3 FOR 2"
1470 PRINT AT 15,4;"BETS REFUNDE
D ON A ""TIE"""
1480 PRINT AT 17,5;"YOU START WI
TH $ 200."
1490 PRINT AT 20,10; PAPER 1; IN
K 9; BRIGHT 1;" GOOD LUCK "
1500 PAUSE 0
1510 GO TO 10
1520 STOP
1530 PRINT  PAPER 6;AT 18,19;" D
O YOU WANT ";AT 19,19;" AN INSUR
ANCE";AT 20,19;" BET? (Y/N)   "
1540 PAUSE 0
1550 IF INKEY$="N" THEN  GO TO 1
740
1560 IF INKEY$<>"Y" THEN  GO TO
1540
1570 GO SUB 1740
1580 PRINT AT 18,19; PAPER 6;"IN
SURANCE BET";AT 20,24;"?"
1590 INPUT IN
```



```
1600 PRINT AT 20,23;"$ ";IN
1610 LET ST=ST-IN
1620 PRINT AT 20,4;ST;"   "
1630 GO SUB 1980: IF V=10 THEN
LET D=21: LET A=18: GO SUB 1790:
 GO TO 1650
1640 RETURN
1650 PRINT AT 20,23;" WINS "
;AT 21,9;"              ": BEEP .
5,20
1660 LET ST=ST+2*IN
1670 PRINT AT 20,4;ST;"   "
1680 LET IN=0
1690 GO SUB 1950
1700 GO TO 1730
1710 PRINT AT 20,23;" LOSES
";AT 21,9;"              ": BEEP 1
,-30
1720 LET IN=0
1730 PAUSE 100
1740 FOR K=18 TO 20
1750 PRINT AT K,19;"
 "
1760 NEXT K
1770 RETURN
1780 GO SUB 1980
1790 FOR C=1 TO 2
1800 PRINT AT B,A;"FJJJG"
1810 FOR N=B+1 TO B+5
1820 PRINT AT N,A;"K   K"
1830 NEXT N
1840 PRINT AT B+6,A;"HJJJI"
1850 NEXT C
1860 BEEP .01,10
1870 IF H=1 THEN  GO TO 1920
1880 GO SUB 2000
1890 PRINT AT B+1,A+1; INK X;A$(
F,4);AT B+3,A+1; INK 0;A$(F,2 TO
 3);AT B+5,A+3; INK X;A$(F,4)
1900 LET A$(F,1)="1"
1910 LET A=A+1: RETURN
1920 FOR N=B+1 TO B+5
1930 PRINT AT N,A;"K"; INK 1;"EE
E"; INK 0;"K"
1940 NEXT N: RETURN
1950 FOR N=1 TO 8: PRINT  BRIGHT
 1; FLASH 1; INK 2; PAPER 6:AT 1
9,10;"▗▘▗▘▗▘▗▘▗▘▗▘▗▘";AT 20,10:"▗▘BL
ACKJACK▗▘";AT 21,10;"▗▘▗▘▗▘▗▘▗▘▗▘▗▘"
: BEEP .2,20: BEEP .2,10: NEXT N
1960 IF P=21 THEN  LET W=INT ((3
*W)/2)/2: GO TO 1170
1970 IF D=21 THEN  GO TO 1130
1980 LET V=0: LET F=INT (RND*52)
+1
1990 IF A$(F,1)<>"0" THEN  GO TO
 1980
2000 LET X=0: IF A$(F,4)="A" OR
A$(F,4)="D" THEN  LET X=2
2010 IF A$(F,2 TO 3)=" A" THEN
LET V=11: RETURN
```

```
2020 IF A$(F,2 TO 3)=" J" THEN
LET V=10: RETURN
2030 IF A$(F,2 TO 3)=" Q" THEN
LET V=10: RETURN
2040 IF A$(F,2 TO 3)=" K" THEN
LET V=10: RETURN
2050 LET V=VAL A$(F,2 TO 3)
2060 RETURN
2070 CLEAR
2080 DIM A$(52,4)
2090 LET A$(1)="0 AA"
2100 LET A$(2)="0 2A"
2110 LET A$(3)="0 3A"
2120 LET A$(4)="0 4A"
2130 LET A$(5)="0 5A"
2140 LET A$(6)="0 6A"
2150 LET A$(7)="0 7A"
2160 LET A$(8)="0 8A"
2170 LET A$(9)="0 9A"
2180 LET A$(10)="010A"
2190 LET A$(11)="0 JA"
2200 LET A$(12)="0 QA"
2210 LET A$(13)="0 KA"
2220 LET A$(14)="0 AB"
2230 LET A$(15)="0 2B"
2240 LET A$(16)="0 3B"
2250 LET A$(17)="0 4B"
2260 LET A$(18)="0 5B"
2270 LET A$(19)="0 6B"
2280 LET A$(20)="0 7B"
2290 LET A$(21)="0 8B"
2300 LET A$(22)="0 9B"
2310 LET A$(23)="010B"
2320 LET A$(24)="0 JB"
2330 LET A$(25)="0 QB"
2340 LET A$(26)="0 KB"
2350 LET A$(27)="0 AC"
2360 LET A$(28)="0 2C"
2370 LET A$(29)="0 3C"
2380 LET A$(30)="0 4C"
2390 LET A$(31)="0 5C"
2400 LET A$(32)="0 6C"
2410 LET A$(33)="0 7C"
2420 LET A$(34)="0 8C"
2430 LET A$(35)="0 9C"
2440 LET A$(36)="010C"
2450 LET A$(37)="0 JC"
2460 LET A$(38)="0 QC"
2470 LET A$(39)="0 KC"
2480 LET A$(40)="0 AD"
2490 LET A$(41)="0 2D"
2500 LET A$(42)="0 3D"
2510 LET A$(43)="0 4D"
2520 LET A$(44)="0 5D"
2530 LET A$(45)="0 6D"
2540 LET A$(46)="0 7D"
2550 LET A$(47)="0 8D"
2560 LET A$(48)="0 9D"
2570 LET A$(49)="010D"
2580 LET A$(50)="0 JD"
2590 LET A$(51)="0 QD"
2600 LET A$(52)="0 KD"
2610 RETURN
2620 RESTORE : FOR a=USR "a" TO
USR "k"+7
2630 READ user: POKE a,user
2640 NEXT a: GO TO 1380
2650 DATA 24,60,126,255,255,126,
60,24
2660 DATA 28,28,8,107,127,107,8,
28
2670 DATA 16,56,124,254,254,254,
16,56
2680 DATA 102,255,255,255,255,12
6,60,24
2690 DATA 204,204,51,51,204,204,
51,51
2700 DATA 0,0,0,7,15,12,24,24
2710 DATA 0,0,0,192,240,48,24,24
2720 DATA 24,24,12,15,7,0,0,0
2730 DATA 24,24,48,240,192,0,0,0
2740 DATA 0,0,0,255,255,0,0,0
2750 DATA 24,24,24,24,24,24,24,2
4
2760 REM a b c d e f g h i j k
2770 REM A B C D E F G H I J K
2780 BORDER 7: PAPER 7: INK 0: C
LS : STOP
2790 CLEAR : SAVE "BLACKJACK" LI
NE 2620
2800 RUN
```

MULTIPLICATION

by John Coffey

```
  60 GO SUB 100: REM Initialize
  70 FOR L=1 TO 50
  80 GO SUB 200: REM Question
  90 GO SUB 300: REM Answer
  95 NEXT L
  98 INPUT "Do you want to play
some more?";a$: IF a$(1)="y" OR
a$(1)="Y" THEN GO TO 70
  99 STOP
 100 REM
 105 REM Initialize
 110 LET HI=100: LET IQ=100
 120 RETURN
 190 REM
 200 REM Question
 210 IF IQ<100 THEN LET IQ=100
 215 IF hi<IQ THEN LET hi=IQ
 220 CLS : PRINT AT 0,16;"IQ =
";IQ-10;"   ";AT 1,16;"Hi =  ";h
i-10
 230 LET Level=(IQ-92)*.25
 240 LET Multi=INT (level*(1-RND
*RND))
 250 LET Multi=INT (level*(1-RND
*RND))
 260 LET b$=STR$ (Multi*Multi)
 270 PRINT AT 4,2;Multi;AT 5,0;"
X ";Multi;AT 5,0; OVER 1;"_____";
AT 11,0;"Type answer before time
 runs out!";"Use spacebar to
                erase wrong answ
ers."
 280 RETURN
 290 REM
 299 REM Answer
 300 LET c$=""
 310 FOR M=1 TO 50
 320 PAUSE 10: LET d$=INKEY$: IF
 LEN d$ THEN GO SUB 400
 330 IF b$=c$ THEN GO TO 500
 340 NEXT M
 350 PRINT AT 5,0;"
AT 5,4-LEN b$;b$"      Time is U
p!"
 351 BEEP .5,-40
 355 PAUSE 500
 360 LET IQ=IQ-4
 370 RETURN
 399 REM Print Answer
 400 IF b$=" " AND LEN c$ THEN L
ET c$=c$( TO LEN c$-1): GO TO 42
0
 410 IF LEN c$<4 THEN IF d$<="9"
 AND d$>="0" THEN LET c$=c$+d$
 420 PRINT AT 5,0;"          "( TO 4-
LEN c$);c$;"  "
 430 RETURN
 499 REM Correct Answer
 500 PRINT "      That's Right!"
 505 BEEP .25,59
 510 PAUSE 100
 520 LET IQ=IQ+1
 530 RETURN
```

Utilities For Zebra Systems' OS-64 Cartridge
by
J. Kevin Paulsen

METAMORPHOSIS

Definition from The Random House Dictionary; entry number three:

metamorphosis n. Any complete change in appearance or CHARacter.

This little Machine Code routine turns the 0S64 character set sideways. The first time it is called it will turn the characters over to the left; the way that you would want them if you were going to write up the left side of your display from position 21,0 up to 0,0. Call it again and the characters will be upside down. Call it a third time and they will appear to be laying over on their right side.

On my 2068 I keep this routine under the BIGCHAR routine taken from the 0S64 Utilities Tape (Zebra Systems). The starting address for this routine is shown in the listing. Use any program you want to POKE the decimal values in from the listing. Be sure to CLEAR 62397. There are 8 bytes set aside for altering the character set. Set the variable for the MC call by LET LAY=62405. After that, all you have to do each time you want to change the character set is use a command like this:

```
10 FLASH USR LAY
```

An alternate method of laying them over to the right is to change the two instructions starting at 62420. To change RLA to RRA you can POKE 62420,31. Change RL (HL) to RR (HL), POKE 62422,30.

A CLOSER LOOK

Now we'll look at the listing.

62405 The form of the CHARacters is stored at 30720 on the 0S64 system.
62408 There are 96 characters to altered.
62410 Temporarily store our count of the CHARacters.
62411 There are 8 bytes that form each CHARacter.
62413 Store this count.
62414 There are 8 bits in each of those bytes to be moved around.
62416 Address 62404 is last byte of those set aside for altering.
62419 Get the byte from the character set.
62420 Rotate the first bit out into the carry flag.
62421 Rotate that bit from the carry flag into our set-aside workspace.
62423 Next byte in workspace.
62424 If we haven't worked all 8 bytes jump back to 62420.
62426 Next byte in character set.
62427 Retrieve the count for the bytes of the character currently being worked on.

62428 If we haven't worked all 8 bytes jump back to 62413.
62430 Next byte in character set.
62431 Set count for 8 bytes to be transferred from our workspace back to the character set.
62434 The last address of the workspace is set for transfer.
62437 Do the transfer.
62439 Calculate where we left off working in the character set.
62445 After the calculation retrieve the count of which character in the set we are working on.
62446 If we've not worked all 96 characters jump back to 62410.
62448 RETurn to BASIC.

Try it out. If you have any questions, suggestions, or comments feel free to write, J. Kevin Paulsen, 216 Mason Ct., Sycamore, IL 60178.

```
62405 LD DE,30720          17,0,120
62408 LD B,96              6,96
62410 PUSH BC              197
62411 LD B,8               6,8
62413 PUSH BC              197
62414 LD B,8               6,8
62416 LD HL,62404          33,196,243
62419 LD A,(DE)            26
62420 RLA                  23
62421 RL (HL)              203,22
62423 DEC HL               43
62424 DJNZ -6              16,250
62426 INC DE               19
62427 POP BC               193
62428 DJNZ -17             16,239
62430 DEC DE               27
62431 LD BC,8              1,8,0
62434 LD HL,62404          33,196,243
62437 LDDR                 237,184
62439 LD BC,9              1,9,0
62442 EX DE,HL             235
62443 ADD HL,BC            9
62444 EX DE,HL             235
62445 POP BC               193
62446 DJNZ -38             16,218
62448 RET                  201
```

DEXTER ET SINISTER

Here are two Machine Code routines that allow you to 'scroll' your 0564 screen to the left and the right. Both routines follow the same basic pattern, so we will describe only the left scroll in detail. Both of them can be moved to any place in memory you like. I've put them in the unused Attribute File for Display File One (DF1). If you are going to POKE the values into memory, just set up your FOR/NEXT loop to READ DATA lines that have the decimal numbers in the listing. There are a total of 89 bytes.

This is the method I use to call these routines. The 0564 does not use the FLASH or BRIGHT commands since there is no Attribute File to alter. The commands have no effect but can still be used in a program line. Two varibles can be set equal to the addresses of the two routines, for example:

```
LET LSLIDE=23096: LET RSLIDE=23142
```

Now you can state FLASH USR LSLIDE which will scroll the screen one column to the left. The USR function returns with a value of 0 so the statement is equivalent to FLASH 0. To scroll to the right state FLASH USR RSLIDE. The variables make it easy to remember how to call the MC and also lets you read your BASIC program without trying to figure out what all those USR addresses mean.

Let's take a look at the LSLIDE listing.

```
23096 The first 7 instructions clear column 0.  Column 0 will
      be partly rolled around to column 63 and we want to be
      sure that we're rolling in a blank column.  Address 16384
      is the first in the Display File.
23099 The next available byte in column 0 is 32 bytes away.
23102 Zero is the number that will give us a clear column.
23104 There are 192bytes down one column to count off.
23106 Clear the byte referred to by HL.
23107 Calculate the next byte to be cleared.
23108 Loop to 23106 until all 192 bytes are cleared.
23110 DF1 starts at 16384.
23113 DF2 at 24576 (all odd numbered columns are stored here).
23116 There are 6144 bytes in each DF to deal with.
23119 Pick up the byte referred to by HL (DF2).
23120 Put it in the byte referred to by DE (DF1).
23121 Make DE refer to DF2.
23123 Make HL refer to DF1.
23125 Step forward to the next byte in DF1.
23126 Pick up the byte referred to by HL (DF1).
23127 Put it in the byte referred to by DE (DF2).
23128 Make DE refer to DF1.
23130 Make HL refer to DF2.
23132 Step forward to the next byte in DF1.
23133 Decrease the counter by 1.
23134 Commit operationsthat will set the zero flag if BC=0.
23136 If BC does not equal 0 the loop back to 23119.
23138 Clear extra byte that crept into the DF.
23141 RETurn to BASIC.
```

Oh! The title??? No, Dexter didn't eat anything bad. It's Latin and means Right And Left. Any other questions, feel free to write me, J. Kevin Paulsen, 216 Mason Court, Sycamore, IL 60178

LEFT SCROLL

```
23096 LD HL,16384          33,0,64
23102 LD A,0               62,0
23104 LD B,192             6,192
23106 LD (HL),A            119
23107 ADD HL,DE            25
23108 DJNZ -4              16,252
23110 LD DE,16384          17,0,64
23113 LD HL,24576          33,0,96
23116 LD BC,6144           1,0,24
23119 LD A,(HL)            126
23120 LD (DE),A            18
23121 SET 5,D               203,234
23123 RES 5,H              203,172
23125 INC HL                35
```

```
23126 LD A,(HL)              126
23127 LD (DE),A              18
23128 RES 5,D                203,170
23130 SET 5,H                203,236
23132 INC DE                 19
23133 DEC BC                 11
23134 LD A,C                 121
23135 OR B                   176
23136 JR NZ,-19              32,237
23138 LD (30719),A           50,255,119
23141 RET                    201
```

RIGHT SCROLL

```
23142 LD HL,24607            33,31,96
23145 LD DE,32               17,32,0
23148 LD A,0                 62,0
23150 LD B,192               6,192
23152 LD (HL),A              119
23153 ADD HL,DE              25
23154 DJNZ -4                16,252
23156 LD DE,30719            17,255,119
23159 LD HL,22527            33,255,87
23162 LD BC,6144             1,0,24
23165 LD A,(HL)              126
23166 LD (DE),A              18
23167 SET 5,H                203,236
23169 RES 5,D                203,170
23171 DEC HL                 43
23172 LD A,(HL)              126
23173 LD (DE),A              18
23174 RES 5,H                203,172
23176 SET 5,D                203,234
23178 DEC DE                 27
23179 DEC BC                 11
23180 LD A,C                 121
23181 OR B                   176
23182 JR NZ,-19              32,237
23184 RET                    201
```

HOORAY FOR HISOFT

by Jack Dohany

The author has no connection with HISOFT other than enthusiastic admiration.

HISOFT is a British firm that produces premium utility software for the Spectrum family of machines. In my opinion, their stuff is the best of its kind. Therefore, I'd like to see their wares get better recognition and distribution in this country.

They also support QL, Amstrad, Amiga, and Atari ST. But in this article I'll focus on their Spectrum products only.

The best way to get HISOFT software is to order it directly from them. Postage to England is 39 cents, and getting what you order usually takes less than 2 weeks. You can pay with personal

check; include extra $$ for currency conversion.
VISA is easier; just say in your letter:
Please charge to my VISA # xxxxxxxxxxxxxxxxxx, Exp: xx/xx/xx.

All HISOFTware will run on 2068 only with Spectrum emulator. Unfortunately, HISOFT are not interested in producing 2068 versions. I feel this could change IF many users from this country were to write them and beg for 2068 versions.

HISOFT did produce a 2068 version of DEVPAC. It had a single bug which I have corrected. They have since WITHDRAWN this version and will not fill orders for it. But I will send it to any legal DEVPAC owners who want it.

HISOFT stuff is all completely in machine-code. It is copyrighted, but NOT copy-protected. The excellent documentation always tells you how to make a backup. The software always has provision for returning to BASIC. HISOFT are continuously improving their software. They have a firm committment to excellence in their products and support.

So, what do they make, and what's so great about it? In this article I can't describe each program fully, but I will list all Spectrum products at the end, and focus today on DEVPAC.
If you have no interest in assembly language, then what follows may have little relevance for you.

****

DEVPAC is, in my not very humble opinion, the most useful assembler/disassembler available for our machines. ZEUS is a toy in comparison.

I believe that even beginning students of assembly language should have the best tools possible on hand. This holds for carpentry, writing, or any craft: weak tools do not serve the novice well.
DEVPAC is two separate programs: GEN (the assembler) and MON (the disassembler/debugger). Each is fully relocatable, that is, it can be loaded at any sensible location in memory. GEN is about 9K long, and MON is about 6K long.

GEN's editor is used to write a "source file". This is a compact ASCII textfile containing labels, Z80 mnemonics and comments. This is also true of ZEUS, and in fact ZEUS's full-screen editor at first seems easier to use than GEN's line editor. But numerous features make the GEN editor actually far more useful than the ZEUS editor.

Once you have written a source file, or loaded one, you can "assemble" that source file into "object code", that is, the desired result of what you are really doing: writing a machine-code program "in English". Here too, ZEUS and GEN do the same thing: translate your source file into object code.
But GEN allows some very useful options during the assembly process which ZEUS does not.

First, there are a whole heap of "special commands" you can put into the source file, analogous to Mscript's format commands. For example, you can load and assemble source directly from tape

during assembly. Thus you can deal with very large source files that would not fit into memory.

Second, you can control where the "output listing" goes during assembly: to the screen or not, to the printer or not, to memory or not, to tape or not. ZEUS has no such thing as an output listing: a grievous omission. Also, with GEN you can control the degree of error-checking, which governs the speed of assembly. You can generate a "dummy" assembly, purely for the purpose of error-checking, without actually generating any object code.

With GEN, in other words, YOU are in full control, and you are not subject to the whim of the person who wrote the program, as you are with ZEUS. I cannot possibly describe all the options allowed by GEN, and their usefulness to the programmer, in this brief article.

****

MON is the DEVPAC disassembler. ZEUS does not include a disassembler; it must be purchased separately.

Like GEN, MON is fully relocatable and can be loaded anywhere there's RAM space for it. You can have GEN and MON in memory at once; they operate independently.

MON has full-screen comprehensive debugging facilities similar to HOT-Z. Because MON was designed for Spectrum and lacks bank-switching facilities, HOT-Z is a better DEBUGGER for the 2068. But it is not a better assembler or disassembler.

The MON disassembler allows you to disassemble an unknown machine-code program into memory as a GEN-compatible source file WITH LABELS. Thus you can "re-source" MC programs... an extremely important capability. Furthermore, you can "exclude" from disassembly as many sections of the code as you like. Excluded sections are not forgotten, but are included as DEFB's rather than as Z80 opcodes. This allows you to make a fairly intelligent disassembly which can then be operated on with GEN.

****

I've written a number of utilities which extend the usefulness of GEN. For example, TEXTMAKER sends GEN assembler output listing to memory in MSCRIPT textfile format; this is how the Customized Mscript source-textfiles were made. Then there's ZG and GZ, to convert Zeus source files into GEN format and vice-versa. If you own DEVPAC and need any of these utilities, just holler.

****

The latest version of DEVPAC as of April 1987 is V4. It has many new features that V3 lacked, including a HELP screen and 51-Column text on screen. The 2068 version of DEVPAC is essentially V3. I plan to adapt V4 to 2068, in the fullness of time, since HISOFT has at present no interest in doing so.

All HISOFT software is microdrive hook-code compatible. The importance of this (particularly for C and PASCAL) may be lost on American users who mostly do not have microdrives, and whose

various mass-storage devices are NOT microdrive hook-code compatible. I know of only one English disc system that's completely compatible: The Disciple.

The importance of such compatibility is this: that ANY software (whether code or BASIC) which accesses microdrive (NOT A&J, but Sinclair) will work equally well with disc or microdrive. And with microdrive, the size of a file is limited not by memory, but by external media (wafer or disc). Thus HISOFT source files (DEVPAC, C or PASCAL) can be simply enormous using microdrive or Disciple disc. This gives our little machines BIG capabilities.

Here please realize that a source file is usually at least 10 times as big as the object code it generates, even without comments. Thus an 8K object file is generated by a source file of at least 80K. Of course large MC programs can be generated in blocks from smaller sources, but this method is tedious and error-prone. That is apparently how the TIMEX ROM was done. The resultant code is not elegant, harmonious or cohesive. It has numerous well-hidden bugs, and it just barely works.

Anyhow, HISOFT is writing stuff as it SHOULD be written, so Hooray for Hisoft! Support (also) your non-local software firm.

HISOFT'S name derives from: HIgh quality SOFTware... and when they coined the name, they lived on High Street. Their current address:

HISOFT
The Old School
Greenfield, Bedford Tel. (0525) 718181
England MK45 5DE Contact: Adam Denning

I picture HISOFT folks as pale reclusive geniuses living and working in a ramshackle abandoned schoolhouse just beyond a small town. In some dusty corner lies a forgotten 2068...

**** HISOFT PRODUCT LIST FOR SPECTRUM *****

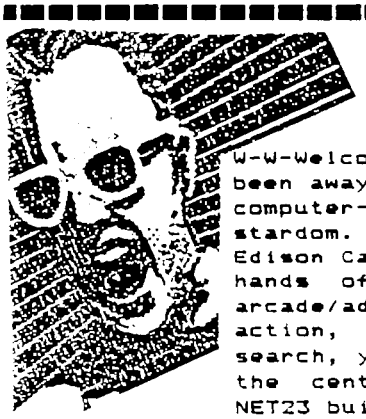
PROGRAM	PRICE (pounds / dollars)
DEVPAC	14.00 / $23.00
-DEVelopment PACkage: a powerful relocatable Z80 assembler and disassembler/debugger.	
C COMPILER	22.00 / $36.00
PASCAL COMPILER	22.00 / $36.00
BASIC COMPILER	22.00 / $36.00
-comprehensive, fast floating-point	
THE COLT	9.00 / $15.00
-a simple fast BASIC integer compiler	
ULTRAKIT	8.50 / $14.00
-a fine BASIC toolkit	

Notes: C and Pascal are high-level languages like BASIC. The

programmer programs in something like English. But C and Pascal are much more flexible and powerful than BASIC.

C and Pascal are not nearly as easy to use as BASIC. In fact I consider both to be much harder to learn than assembly language. C and Pascal are for very intelligent and serious programmers. That leaves me out; how about you? I'm sure there MUST be some Sinclair folks out there who are bright enough to use C and Pascal effectively. C and Pascal programs run much faster than BASIC programs, though not as fast as "pure" machine-code. They also take up quite a bit of "minimum-room" in memory, though a big C or Pascal program would ultimately take up LESS room than an equivalent BASIC program.

Adapting C or Pascal to run on 2068 is something I contemplate with nothing but fear and loathing. HISOFT could do it with relative ease, if they were motivated to do it. So let's motivate them! Rah Rah Zis Boom Bah! Let 'em know we're here!



MAX HEADROOM

by Michael E Carver

W-W-Welcome to the world of M-M-M-Max Headroom! In case you have been away from the planet for a while. . . Max Headroom is a computer-synthesized TV announcer, who has gained world-wide stardom. As the player, you take on the role of TV newshound, Edison Carter, in his quest to retrieve the Max Module now in the hands of Network 23. The game is billed as another arcade/adventure, though most of the emphasis is on the arcade action, with only a smattering of adventure. To aid in your search, you have access to the Maxhunter, which is tapped into the central computer, controlling operations and security in the NET23 building.

Your first task is to gain access to the top floors (floors 200-211) via the lift (elevator) system. Various memos have been lifted from the NET23 computer, to aid in your quest. These memos form the instructions included on the inlay provided. Due to a newly installed security system, this can be difficult to achieve. Once the lift system has been mastered, one must pass another security test to gain access to the offices on each floor, where various codes are located. To hamper your search, each floor is guarded by armed robots plus the vicious thugs, Breugel and Mahler. Control is via keyboard (which is definable) or joystick. I found control from the keyboard easier, as some tasks involve quick and accurate control of a cursor. The joystick seemed to be sloppy and inaccurate.

A portion of the screen is devoted to a 3-D view of the action. The graphics are adequate, but are basically line drawings of office walls, desks and characters. Once the main objective of the game is accomplished, one can load the second portion of the game. This part is an animated Max Headroom, providing a victory speech via synthesized speech, using the beep command. Though it is fasinating to hear that little beep try to speak, it is also a bit difficult to understand.

If you are a big Max fan, you will probably wish to give this game a try. I found myself continually trying to reach the final goal. The last puzzle left me baffled for quite some time. Even though the action is very repetitive, I strived for the final glory. Once the game is completed, the challange is gone, and I doubt many would find themselves going back to it very often.

Max Headroom is available from Curry Computer for $16.95.

Orders to Curry Computer can be mailed to PO Box 5607, Glendale, AZ 85312-5607, or by calling 1-800-628-2828 ext. 950 24 hours a day and 7 days a week.

LIGHT PENS

by Duncan Teague

In a previous article I summarized the pros and cons of the different means of Spectrum emulation. One of the emulators, the Rainbow Plus Interface, has a singular advantage. Its Spectrum through-connector allows the use of Spectrum peripherals on a T/S 2068. One of the Spectrum peripherals I used to test this feature of the Rainbow Plus was the DK'Tronics Light Pen. Since I also own the Zebra Systems Light Pen, I'll compare the two.

One feature light pens have in common is the necessity of calibrating them to function properly with your system. The procedure is simple. You point the pen at a designated point on the screen, watch the screen for feedback from the computer, and them make "adjustments" as necessary until the readings are optimum for your system.

During this procedure, the computer, the software, and the light pen work in concert to take readings from the screen. They measure the timing of the beam of light that sweeps across the screen to make the picture. The precise time the beam of light is detected by the light pen tells the computer the coordinates of the point on the screen.

The intensity of the light and the contrast between light and dark on your screen affect the function of the light pen. Once the light pen is calibrated, the brightness and contrast of the screen should not be changed.

Both light pens plug into an interface which in turn plugs into the computer. The interface for the DK'Tronics unit attaches to the Rainbow Plus on 2068's or directly into the edge connector on Spectrums. The interface for the Zebra Light Pen plugs into the cassette input port. The major difference between the two devices is best illustrated by the sophistication of the accompanying software.

The demonstration program with the Zebra Light Pen includes two routines, both of which use the light pen as a substitute for the keyboard to make "menu" type selections from the screen. The pen is pointed at a flashing square. This is comparable to using the INKEY$ function with the keyboard. Resolution is thus one character cell and reliability is about 85%.

The DK'Tronics software is a drawing program capable of one pixel resolution. Thus, theoretically, the resolution of the DK'Tronics Light Pen is (8*8=) 64 times that of the Zebra Light Pen. In practice it is very difficult to aim the blunt end of the light pen with any precision.

The DK'Tronics drawing program has sixteen functions. They include routines for designating "origin" and "target" points for drawing precise arcs, circles, lines, and rectangles; sketching free-form figures; "filling" enclosed areas; and erasing what you've drawn. Other routines are provided for adding text from the keyboard, changing screen color attributes, storing to and recalling screens from both memory and tape, and clearing the work area.

The light pen is used to select routines from a menu at the bottom of the screen, below the work area. The menu is comprised of sixteen squares, above which appear the initial letters of the various functions. When using the routines, a thick black band, which "determine[s] the position of pen," obscures the part of the screen on which you want to draw. This artifact of the program's operation combined with the difficulty of accurate positioning of the light pen make drawing a Herculean task.

The documentation for the Zebra Light Pen is as limited as the software. There is, however, a BASIC program listing of the calibration procedure, and the demonstration program is all in BASIC and available for study.

The manual for the DK'Tronics Light Pen is more thorough. It includes instructions for saving both the "menu selection" and the "drawing routine" portions of the software's machine code for use within your own BASIC programs.

I found the Zebra Light Pen worked better on a black and white television set than on a color monitor. The reverse is true of the DK'Tronics Light Pen. Both have a common drawback. I find that my arm gets tired from constantly holding up the light pen to the screen.

Both showed promise, but both fall short of being practical tools. The Zebra Light Pen software is limited in its scope. The DK'Tronics Light Pen software is more ambitious, but it is difficult to obtain desirable results with the drawing program. It would be difficult for me to recommend either for other than experimental purposes or as a novelty.

DK'tronics Ltd., Unit 6, Shire Hill Industrial Estate, Saffron Walden, Essex CB11 3AQ, U.K.

Zebra Systems Inc., 78-06 Jamaica Ave., Woodhaven, NY 11421

FAIRLIGHT

(a program by Bo Jangeborg)

reviewed by Michael E. Carver

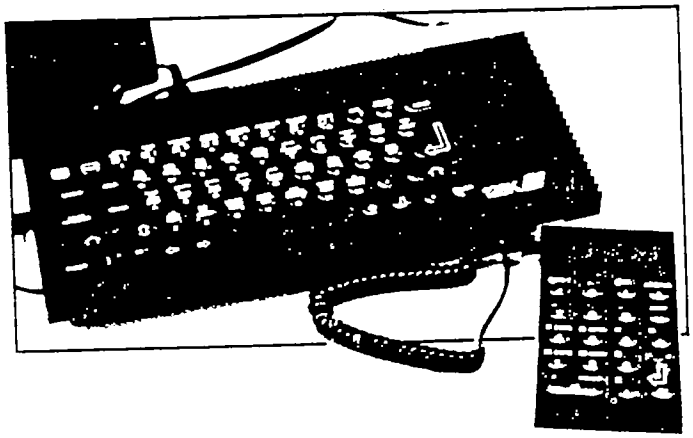
This game is one of the best of the Arcade/Adventure genre. Fairlight was written by the author of the Artist (see March/April 1986 Time Designs). It is a 3-dimensional real-time adventure. The method used to create this animated world is described as "The 3-D Worldmaker Technique". The graphics were drawn with an updated version of the Artist (which is now available as Artist 2). To avoid attribute bleeds, all screens are monochrome, with separate colors used for different areas of the game world. The world created with this technique is quite impressive. Most of the objects encountered can be pushed, picked-up, or stacked. Each item has its own characteristics (weight, momentum and sometimes intelligence). As the player, you manipulate the character Isvar and are allowed to carry up to five items, depending on weight.

The goal of the adventure is to locate the "Book of Light", which was hidden in the Castle Avars. The castle is protected by magic and magical beings. By locating the book and presenting it to the imprisoned sorcerer, magic will be restored to the world. Of course, you are trapped in the castle and returning the book is your only hope of escaping. The castle contains several levels of rooms, courtyards and, of course, dungeons and caves -- all of which are wonderfully presented in a 3-D view. The action is viewed from a "ceiling-corner-view". Control is via keyboard (groups of keys for movement, fighting, manipulating objects, etc.) or with a Kempston-compatible joystick. Keyboard control is quite responsive, though confusing at times, as the character moves diagonally across the screen/room.

As Isvar explores the castle in search of the Book of Light, he must fight guards, trolls and avoid "magical" beings, which cannot be fought. The fighting is done by sword stabs, and the action of Isvar and the guards is enjoyable to watch. The puzzles encountered range from easy to difficult (some of which I am still struggling with). There are hidden rooms and items which have some magical use. As Isvar battles or comes in contact with deadly magical forces, his strength is decreased and he must locate food or magic elixirs to restore his strength. Many hours will be spent just exploring the castle and its contents, while trying to survive the "traps". Many more will be spent trying to put all of the pieces together.

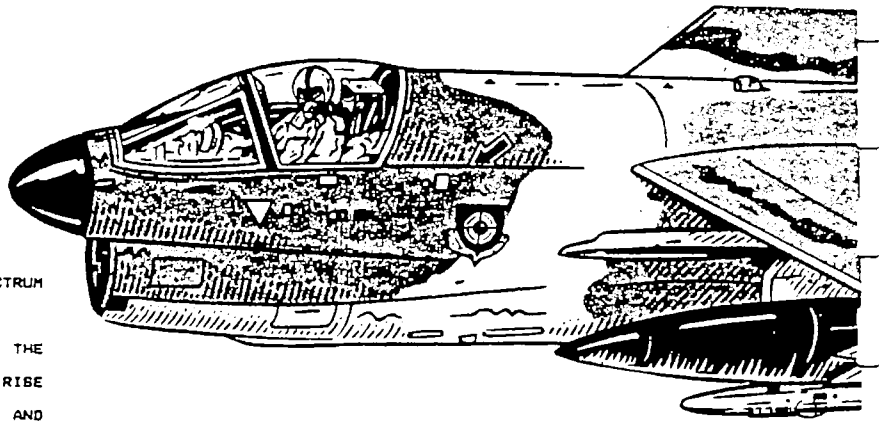
Fairlight is a very enjoyable package, a must for those who like a good mix of adventure with their arcade action. The loading screen is one of the best pieces of art I have seen produced with the Spectrum. Along with the opening music (via beep), it is well worth loading, just for this introduction. There are a few minor drawbacks with the program. The first is common with all such type games -- the slowing down of action when many objects are animated at one time on the screen. The other problem is more a result of the fine detail in the graphics. A few items may not be easily recognized, unless one is using a monitor.

The program is packaged in a pocket sized plastic case with a booklet explaining the "rules" and a brief story to set the stage. The program is available from Curry Computee for $17.95.



SPITFIRE 40

by Dennis Silvestri



SPITFIRE 40 IS A FLIGHT SIMULATION PROGRAM FOR THE SPECTRUM EQUIPPTED T/S 2068 COMPUTER.

THIS PROGRAM SIMULATES THE FLYING AND COMBAT TECHNIQUE OF THE FAMOUS WW2 BRITISH FIGHTER PLANE. YOUR ULTIMATE GOAL IS TO RISE THROUGH THE RANKS TO ACHIEVE THE TITLE OF GROUP CAPTAIN AND EARN VARIOUS MILITARY MEDALS. THERE ARE THREE AREAS TO CHOOSE FROM, PRACTICE, COMBAT PRACTICE, AND FULL COMBAT. EACH AREA HAS VARIOUS SKILL LEVELS THAT YOU CAN ADVANCE TO.

IN THE PRACTICE MODE YOU ARE GIVEN THE OPPORTUNITY TO GET FAMILIAR WITH FLYING THE AIRCRAFT. YOU CAN TAKE OFF AND LAND AT DIFFERENT AIRFIELDS AND GENERALLY GET THE FEEL FOR FLYING.

COMBAT PRACTICE ALLOWS YOU TO MANEUVER THE AIRCRAFT IN A COMBAT SITUATION, IN WHICH YOU CAN SHOOT DOWN ENEMY PLANES WITHOUT BEING SHOT DOWN BY RETURN FIRE. FULL COMBAT IS WHERE YOU SCRAMBLE THE SPITFIRE FROM AN AIRFIELD AND GO ONTO INTERCEPT ENEMY AIRCRAFT. IN THIS MODE ENEMY AIRCRAFT RETURN FIRE.

IN THE PRACTICE AND FULL COMBAT MODE YOUR FLYING TIME AND THE NUMBER OF VICTORIES ARE RECORDED AND CAN BE SAVED, BUT ONLY IF YOU RETURN AND SAFELY LAND AT AN AIRFIELD. THIS CAN BE LOADED BACK INTO THE MAIN PROGRAM AND ALLOWS FOR THE BUILD UP OF VICTORIES AND FLYING TIME WHICH ARE REQUIRED IF ONE IS TO REACH THE ULTIMATE GOAL.

UNLIKE OTHER FLIGHT SIMULATION PROGRAMS THAT GIVE INSTRUMENT READINGS AND A VIEW OUT THE COCKPIT ON ONE SCREEN, SPITFIRE 40 HAS ONE SCREEN FOR A VIEW OF INSTRUMENTS ONLY WHICH INCLUDE: FUEL GAUGE, AIR SPEED, ARTIFICIAL HORIZON, ENGINE RPM, VSI, SLIP AND TURN, COMPASS, ALTIMETER, RUDDER INDICATOR, FLAPS, LANDING GEAR, BRAKES, AND PITCH INDICATOR AND A SEPARATE SCREEN FOR A VIEW OUT THE COCKPIT.

WITH THIS PROGRAM I FAVOR THE SEPARATE SCREENS. THE INSTRUMENT VIEW IS LARGER, MORE CLEARER, AND EASIER TO READ. THE VIEW OUT THE COCKPIT IS ALSO LARGER AND CLEARER. SWITCHING BETWEEN SCREENS CAN BE DONE AT ANY TIME WITH THE SPACE BAR. THERE IS ALSO A THIRD SCREEN WHICH GIVES THE AIRCRAFTS POSITION RELATIVE TO THE GROUND AND TO ENEMY AIRCRAFT.

THE COLOR GRAPHICS ARE VERY GOOD, ESPECIALLY WHEN ENEMY AIRCRAFT ARE IN FRONT OF YOU, THEY CLIMB, DIVE, BANK, TURN, AND EVEN ROLL OUT TO AVOID BEING SHOT DOWN AND THEY ARE CLEARLY VISIBLE AS AIRCRAFT. THERE IS EVEN A REAR VIEW MIRROR IN THE COCKPIT VIEW IN WHICH ENEMY AIRCRAFT APPEAR WHEN THEY ARE ON YOUR TAIL.

SOUND GENERATION IS ALSO GOOD, BEING SUPPLIED BY MACHINE GUN FIRE, ENGINE REVS AS SPEED IS INCREASED OR DECREASED, EVEN THE CHIRPING OF TIRES ON THE GROUND AS YOU LAND AND APPLY THE BRAKES. SOME UNUSUAL SOUNDS OCCUR IF YOU ARE SHOT DOWN OR CRASH INTO THE GROUND. THIS PROGRAM USES JOYSTICK CONTROL AS WELL AS THE KEYBOARD, AND A KEMPSTON OR KEMPSTON COMPATIBLE INTERFACE IS REQUIRED TO USE A JOYSTICK.

SPITFIRE 40 IS DEFINITELY WORTHY OF PERSEVERANCE AND MOST REALISTICALLY GIVES YOU THE SENSE OF BEING IN THE COCKPIT. IT COMES WITH COMPLETE AND EASY TO UNDERSTAND DOCUMENTATION WHICH ALSO INCLUDES A BRIEF HISTORICAL BACKGROUND ON THIS FAMOUS AIRCRAFT. ALSO INCLUDED IS INFORMATION ON THE THEORY OF FLIGHT.

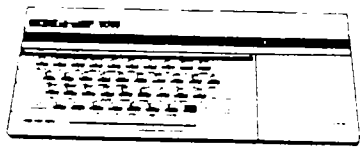
THIS PROGRAM IS AVAILBLE FROM SPEEDYSOFT FOR £ 9.95 PLUS £ 2.00 SHIPPING. (THIS IS BRITISH POUNDS) APPROXIMATELY $ 18.00 U.S. DOLLARS WHEN I ORDERED IT.

ANYONE WITH A SPECTRUM EQUIPPED 2068 WHO IS NOT FAMILIAR WITH SPEEDYSOFT SHOULD SEND FOR THEIR CATALOG. THEY HAVE MANY PROFESSIONAL AND GAMES PROGRAMS TO CHOOSE FROM. THEY ALSO GIVE SUPERIOR SERVICE. THEIR COMPLETE ADDRESS IS:

SPEEDYSOFT

THE BLACKSMITHS, RADNAGE COMMON, BUCKINGHAMSHIRE

HP 14 4DH, ENGLAND



QL CORNER

SINCLAIR QL

by M. Vincent Lyon

INTRODUCTION

QL CORNER was originally a series of columns written by Vince Lyon and published in "The Plotter", the newsletter of the Clackamas County Area Timex Sinclair Users Group of Oregon. The series was first introduced in March 1986 and ran until the December 1986 issue.

We have reprinted the series in part here with generous permission from the author. The original layout has been slightly re-arranged and edited to fit into a tutorial lesson format. The series is an excellent introduction and study of QL programming concepts. - T. Woods

LESSON NUMBER ONE

It may seem logical that the best place to begin a tutorial on the QL, would be a simple explanation of the basic differences between the QL and other Sinclair computers. Since, however, the QL is based on a new chip with new programming concepts and commands, the QL does not even look like a distant cousin of the 2068, nor appear to come from the same galaxy as the ZX81. There is, simply no comparison.

A quick look at the program below clearly demonstrates this new, unique personality on the QL. As you will notice, the main loop of the program is only four lines long, and the majority of the remainder is a collection of defined procedures and functions that are called by name from the loop.

This ability to define any procedure and call it at will, by name is truly a Quantum Leap over 2068 BASIC. It is, to me, the single most exciting addition to the QL.

Although the program isn't meant to do anything fantastic, it does serve to demonstrate a few of the new commands that are available to the programmer. In the example "FILLS" fills window #1 with periods which serve no purpose other than to prove that as one moves the cursor around, the periods are not erased.

The remainder of the program prints a cross-hair cursor on the screen which can be moved about by use of the cursor keys. The keys are read in "keyread", the position is updated in "position" and the cursor drawn in "lne" (since LINE is a QL command, it could not be used).

Let's look at a few of the new commands used in this program:

OVER-1 - This command reverses the pixels at the print location. In this example, it permits the cursor to move about without overwriting screen data.

LINE - No longer a DRAW command, LINE permits a user to draw a line from one specific pixel location to another.

LOCal - Allows a user to set up a variable for use ONLY within a DEFined procedure. If the same variable is used within the main program, it will not alter the variable in the DEFined procedure (and vice-versa).

THEN RETurn x - A nifty, fast way to say if x=1 then let y=1.

Well, those are a few of the new commands used in this short sample program, and there are many more yet to explore (at least 25). These few examples, however, clearly help me to establish the concept that the QL is indeed unique, with little that appears related to the 2068. If one were to wish to convert this program to run on the 2068, it would be at least twice as long and much slower.

If you have a QL, enter and test the program. If you have a TS2068, try to convert it.

```
100 LET a=50:b=50:s=0
110 PAPER 0:=INK 7:BORDER 0,0:CLS
120 PRINT FILL$ (",",665)
130 lne:disp:wprint
140 REMark ------MARK LOOP
150 REPeat loop
160 Position
170 IF s=1 THEN EXIT loop
180 END REPeat loop
190 REMark -----END MAIN LOOP---
200 STOP
210 REMark ----------------------
220 DEFine PRICedure lne
230 REMark ----------------------
240 OVER-1
250 LINE a,b TO a+10,b:LINE a+15,
    b TO a+25,b
260 LINE a+12,b-12 TO a+12,b-2:LINE
    a+12,b+2 TO a+12,b+12
270 OVER 0
280 END DEFine
290 REMark ----------------------
300 DEFine FUNction keyread
310 REMark ----------------------
320 LOCal q$
330 REPeat loop
340 LET q$=INKEY$(-1)
350 IF q$=CHR$(192) THEN RETurn 1
360 IF q$=CHR$(200) THEN RETurn 2
370 IF q$=CHR$(208) Then RETurn 3
380 IF q$=CHR$(216) THEN RETurn 4
390 IF q$=CHR$(248) THEN RETurn 5
400 END REPeat loop
410 END DEFine
420 REMark ----------------------
430 DEFine PROCedure Position
440 REMark ----------------------
450 LOCal x
460 REPeat options
470 x=keyread
480 SELect ON x
490 =1:lne:LET a=a-1:lne:wprint
500 =2:lne:LET a=a+1:lne:wprint
510 =3:lne:LET b=b+1:lne:wprint
520 =4:lne:LET b=b-1:lne:wprint
530 =5:LET s=1:RETurn
540 END SELect
550 END REPeat options
560 END DEFine
570 REMark ----------------------
580 DEFine PROCedure disp
590 REMark ----------------------
600 WINDOW #2,110,10,200,175
640 DEFine PROCedure wprint
650 REMark ----------------------
660 PRINT #2," ";a;" ";b
670 END DEFine
```

LESSON NUMBER TWO

You probably would like to write some special programs for your QL. Maybe even a space arcade or adventure game. BUT, designing the screen is a real chore.

Suppose you wanted the screen to display everything the pilot would see: 1) a center video display which can face fore or aft on command, 2) an overhead console which reports on weapon status, 3) a small panel on the right center which shows aiming reports and target range, and 4) a below screen panel which reports on fuel, range from base, life support systems, etc.

If you have anything besides a QL, then you better get out your manual on Machine Code. With the multi-tasking QL, however, this screen display is a real snap.

On the QL one can set up all the windows in advance (we could use a DEFine PROCedure), and can print to any window and data or calculation result. Each window can be a color, paper and ink, a different size with a different border, and can print in character sized and designed strictly for it.

The program below shows some simple window designs with different colors and borders. I didn't use the full range of all the available character sizes, because I am sure you will want to experiment with the defined windows.

```
5 MODE 4:WINDOW #1,500,250,0,0,:CLS
#1
10 WINDOW #1,440,150,30,0,:BORDER
#1,1,7:CSIZE #1,2,0:PAPER #1,1,:CLS
20 WINDOW #0,440,30,30,155:PAPER
#0,0:CSIZE #0,0,0:PAPER #0,5:INK
#0,0:BORDER #0,1,7:CLS #0
30 CLS #1:CLS #0
40 INPUT #0;"enter your name ";z$
50 PRINT "HELLO ";z$
52 PAUSE 120
53 PRINT \\\\"GOODBY ";z$
54 PAUSE 120
55 MODE 4:WINDOW #,500,250,0,0:CLS
#1
60 BORDER #1,0,0:CLS
70 WINDOW #1,440,100,30,0:PAPER
#1,2:BORDER #1,1,3:CSIZE #1,2,0:CLS
80 WINDOW #0,440,100,30,105:PAPER
#0,0:INK#0,7:CSIZE #0,0,0
90 INPUT #0;"Enter your name ";z$
100 PRINT "HELLO ";z$
110 PAUSE 120
120 PRINT \\\"GOODBY ";z$
130 PAUSE 120
135 MODE 4:WINDOW #1,500,250,0,0:CLS
#1
140 WINDOW #1,200,100,30,0:BORDER
#1,1,7:CSIZE #1,2,0
150 WINDOW #2,230,100,240,0:PAPER
#2,0:BORDER #2,1,7:CLS #2
160 WINDOW #0,440,75,30,100:BORDER
#0,1,7:PAPER #0,2:INK #0,6:CSIZE
#0,0,0:CLS #0
170 INPUT #0\\"Enter your name ";z$
175 PRINT #0\\\\\\\\\\
180 PRINT "well ";z$\\"meet my
windowe"\\"this is #1."
185 PRINT #2;"this is window #2"
190 PRINT #0;"and this is window #0
(the input window)"
```

As you may note, Line #5 sets up a window which is re-designed by Line #10. That is to make sure the screen is cleared of all previously defined windows. The rest of the program is just a demonstration of printing windows of different sizes at different screen locations. Remember, that window #0 is always the input window. When the program ends, the cursor will be in that window.

If you don't want to write the world's greatest space game, you will still find useful applications for windows in business and game programs.

LESSON NUMBER THREE

I have heard from some individuals that the programs thus far printed did not appear on the screen as intended. Since most of the QLs out there are running on TVs and not monitors, all the programs have been designed for a TV display. Before you enter this program, make sure that you are in TV mode. You can always re-design the program if you wish.

This months program is just a fun little demo of the turtle graphics. As anyone who has experimented with turtle graphics can attest, this isn't a true example of the full capabilities of that turtle. But, it does serve to demonstrate the basic commands available on the QL.

Since by now, you all know how to use DEFined PROCedures, I put all the turtle commands in a PROCedure called "doit". All the print locations and ink colors are contained within the DATA statements. In using trutle graphics as designed for the QL, it is nice to note that the commands POINT and TURNTO may be variables, calculated from within the program or assigned by a DATA and READ statement.

```
10 PAPER 0:CLS:CLS #0
20 FOR m=14 TO 18 STEP 2
30 RESTORE : PAUSE 60: CLS
40 FOR x=1 TO 5
50 READ a,b,c
60 doit
70 DATA 30,27,1,80,27,2,130,27,3
80 DATA 55,72,4,105,72,5
90 NEXT x
100 NEXT m
110 a=80:b=54:c=7:m=30: doit
120 FOR z=1 TO 7:c=z:doit:NEXT z
130 RECOL 0,2,0,2,0,0,0,6
140 PRINT #0\\,"That's all folks ";
150 PAUSE 240: RUN
160 DEFine PROCedure doit
170 POINT a,b
180 INK c
190 PENDOWN
200 TURNTO 0
210 FOR k=1 TO 28
220 MOVE m:TURN 90
230 MOVE m:TURN 90
240 MOVE m:TURN 90
250 MOVE m:TURNTO m*k
260 NEXT k
270 END DEFine
```

LESSON NUMBER FOUR

One of the new commands available to the QL user is SELect. At first glance, it may not appear to be much of an addition, but, as the program below demonstrates, there is more to the command than meets the eye.

First, remember that from within a SELect, it is possible to revalue any variable, call any defined procedure or function or any task you may want to assign. It can work from keyboard input, or the result of computations whithin a program (IF X=1 Then).

In the sample, it simply changes the p string and calls a procedure (the same procedure for all), but it could just as easily call a separate procedure for each SELect, or two or more procedures. An inventive basic programmer can find a multitude of possibilities here for fast and easy procedure calls.

```
20 WINDOW #0,450,40,35,150
22 CSIZE #0,1,0
24 BORDER #0,5,7
26 CLS:CLS #0
28 mainmenu
30 INPUT #0\\"    SELECT A NUMBER: ";fi$
32 errtrap
34 fi=fi$
36 findit
38 STOP
40 DEFine PROCedure mainmenu
42 CLS:CLS #0
44 PRINT\\\\"  SELECT TEST"\\
46 RESTORE 54
48 FOR count=1 TO 6
50 READ m$:PRINT TO 5,m$\
52 NEXT count
54 DATA "1","2","3","4","5","6"
56 END DEFine
58 DEFine PROCedure findit
60 SELect ON fi
62 =1:p$="one":printit
64 =2:p$="two":printit
66 =3:p$="three":printit
68 =4:p$="four":printit
70 =5:p$="five":printit
72 =6:p$="six":printit
74 END DEFine
76 DEFine PROCedure errtrap
78 IF CODE (fi$)<49 THEN redo
80 IF CODE (fi$)>54 THEN redo
82 END DEFine
84 DEFine PROCedure redo
86 GO TO 28
88 END DEFine
90 DEFine PROCedure printit
92 CLS:CLS #0
94 PRINT\\\\ TO 4,"This is "&p$
96 PRINT #0\\,,"PRESS ANY KEY ";
98 PAUSE
100 CLS:CLS #0:redo
```

LESSON NUMBER FIVE

Recently I read a book which was supposed to be on SuperBASIC for the QL. However, I was more than a little surprised when I came to the section on graphics. It seems that the author thought the graphics potential of the QL was best utilized by loading the included software and reading your manual on QL Easel.

While that may be well and good for those select few who don't wish to include charts or graphs within their own programs, it certainly is not for those who want to design real business programs.

It is not a great chore to use the graphics functions available from BASIC to create charts and graphs. And, the speed of the QL, makes it possible to use charts more quickly from within your own programs than it is to stop and load Easel.

Rather than explain in detail the entire program below, I'll let it stand on it's own merit. You will find it easy to include in your own programs.

```
10 MODE 4:WINDOW #1,450,165,35,0:
PAPER #1,0:INK #1,7:BORDER #1,1,7:
CSIZE #1,0,0: CLS
20WINDOW # 0,450,25,35,167: PAPER
#0,2: INK #0,7: BORDER #0,1,5:
CLS #0
30 high = 0: low = 100: ac= 0
40 UNDER 1: STRIP 3
50 FOR k = 0 TO 19
60 LET lth = RND (2 TO 65)
70 ac = ac+lth
80 IF lth>high THEN LET high = lth
90 IF lth<low THEN LET low = lth
100 AT k,0: PRINT " "; TO lth;"   ";
lth
110 NEXT k
120 PRINT #0,,,,"Horizontal Bar
Chart"\,,"High = ";high;"   Low = ";
low, "Average = ";(ac/20)
130 FOR k = 10 TO  STEP 10
140 LINE k,0 TO k,175
150 NEXT k
160 PAUSE
170 UNDER 0: STRIP 0
180 CLS: CLS #0
190 av = 0
200 FOR k = 1 TO 55
210 LET hgt = RND (5 TO 150)
220 av = av + hgt
230 BLOCK 5, hgt, k*8,160-hgt, 5
240 NEXT k
250 PRINT #0,,,, "Vertical Bar
Chart"
260 PRINT #0,,,, "Average =  ";
(av/43)
270 PAUSE
280 CLS: CLS #0: POINT 0,0
```

```
290 AC = 0: CNT = 0
300 FOR K = 0 TO 170 STEP 2
310 LET np = RND (5 TO 100)
320 ac = ac + np:cnt = cnt + 1
330 LINE TO k,np
340 NEXT k
350 INK 2: OVER -1
360 For k = 0 TO 170 STEP 5
370 LINE k,0 TO k,175
380 NEXT k
390 FOR k = p TO 100 STEP 5
400 LINE 0, k TO 430, k
410 NEXT  k
420 INK 7: OVER 0
430 PRINT #0\,,,, 'Line Graph'\,,,'
    Average = ';ac/cnt
440 LINE 0, ac/cnt TO 430, ac/cnt
450 PAUSE
460 CLS: CLS #0
470 PRINT #\\,,,,'End Of Demo'
480 PAUSE
490 CLS #0
```

LESSON NUMBER SIX

To run the following demo, have your QL in TV mode and a formatted cartridg in MDV1_.

Now we have a chance to look at a few of the facilities and commands we haven't explored before. Note line 35 (a%=K) which is the new QL method of assigning the INT command. When the program is run, you will notice that the QL always rounds properly without adding the .5 to the function as was necessary with the 2068.

The program then will do some demos of the PAN and SCROLL commands, and a screen save and recall. If you run the program a second time, make sure to DELETE MDV1_display, since the QL will not automatically overwrite an existing file.

```
10 DEFine PROCedure show_int
15 PAPER 0:INK 7:BORDER 6,2,7
20 CLS:CLS #0
25 PRINT:PRINT
30 FOR k=1 TO 2 STEP .1
35 a%=k
40 PRINT TO 3;"k=";k;
45 INK a%+2
50 PRINT TO 2+(a%*10); "INT = ";a%
55 INK 7
60 END FOR k
65 PAUSE 240
70 END DEFine
75 DEFine PROCedure move_up
80 FOR k=1 TO 105
85 SCROLL -1
90 NEXT k
95 CLS
100 END DEFine
110 DEFine PROCedure move_down
115 FOR k=1 TO 135
120 SCROLL 1
125 END FOR k
130 CLS
135 END DEFine
140 DEFine PROCedure move_right
145 FOR k=1 TO 165
150 PAN 3
155 END FOR k
160 CLS
165 END DEFine
170 DEFine PROCedure do_display
175 CLS:CLS #0
180 BLOCK 70,130,10,10,5
185 BLOCK 300,10,65,70,2
190 BLOCK 20,120,35,15,3
195 AT 5,8:PRINT "THIS IS THE DISPLAY"
200 AT 12,10:PRINT "I WILL SAVE IT"
205 AT 15,15:PRINT "TO MDV1_"
210 END DEFine
215 DEFine PROCedure file_display
220 SBYTES mdv1_display,131072,32768
225 PRINT #0\\,,"NOW IT'S SAVED"
230 END DEFine
235 DEFine PROCedure recall_display
240 CLS #0
245 PRINT #0\\,,"I'M READY TO RECALL"
250 PAUSE 240
255 PAPER 0
260 BORDER 0,0
265 CLS:CLS #0
270 LBYTES mdv1_display,131072
275 END DEFine
300 show_int:move_up
305 show_int:move_down
310 show_int:move_right
315 do_display:file_display
320 PAUSE 360:CLS
325 recall_display
330 PRINT #0\\,"    END OF DEMO ";
```

A NOTE ON ARCHIVE

Besides exploring aspects of SuperBASIC, M. Vincent Lyon has also established himself as somewhat of an authority on ARCHIVE...perhaps the most misunderstood program of the Psion suite. Mr. Lyon has written a popular book on the subject called ARCHIVE MASTER. His book covers many uses and applications of the program, and takes the user through all of the commands and functions. In the process, the user will end up with some valuable business software like inventory, accounts receivable, and mail list programs. ARCHIVE MASTER recently received very high marks in a review published by a national PC magazine. If you would like to get details about the book, contact: RMG ENTERPRISES, 1419 1/2 7th Street., Oregon City, OR 97045, phone (503) 655-7484. -T. Woods

QUILL Update For The Sinclair QL Printer

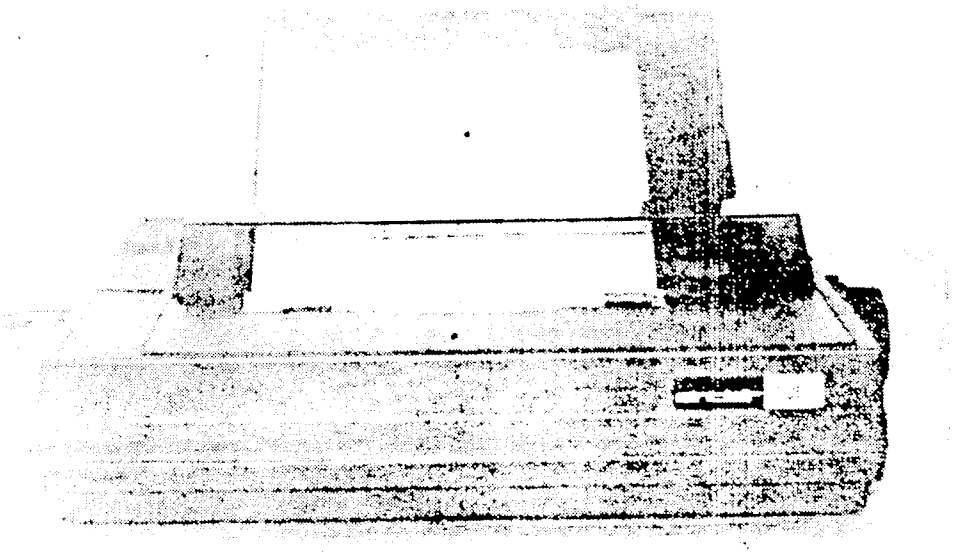
by Matthew Zenkar

To adapt Quill to drive the Near Letter Quality mode of the Sinclair QL printer, Place your backup Quill cartridge in Microdrive 1 and, type **lrun mdv1_INSTALL_BAS ENTER**.
When install_bas is through loading, it will be running and, you will be prompted to enter 1, or 2, for the Microdrive on which you wish to install the new printer. Type **1 ENTER**.
The next prompt will ask you to indicate which type of port you are using serial or parallel. Press **ENTER** for the serial port.
When the printer list appears press either ↑↓, until the cursor reaches the printer driver entitled **OTHER**. Once the cursor is on OTHER press **F2**. The effect is to copy the OTHER printer driver, to allow you to rename it and, set up a new printer driver. After you are through setting up the new printer driver, the OTHER printer driver will still be on the list as will the new one. OTHER cannot be deleted.
After pressing F2, a list will appear like the ones on pages 4 & 6, of the information section of your QL manual. Press → then type **QL-NLQ ENTER**. This gives the new printer driver a name.
Press ↓ until you reach the line CONTINOUS FORMS: Then press →. This will change the word NO to YES. The QL Printer handles continous forms.
Press ↑ until the cursor moves up to the name line and hold it. If you are using a TV, this will shift the list up so that the first line displayed is, the END OF LINE CODE: line. Keep the ↑ key depressed until, the cursor reaches the Preamble code line. The end of line code can be left alone. The nemonics CR,LF have the same numeric equivalents to the nemonics CR,NL which is the code that the QL Printer uses for the end of line code.
PREAMBLE CODE: line. Press → then type exactly, **ESC,'@,CAN,ESC,'x,'1 ENTER**. After you have pressed enter, the line must look exactly like the line on the back of this sheet. If it does not, press → again and repeat the procedure. To enter nemonics like ESC you need only type **ESC** followed by a comma(,). To enter numebrs or letters they must be preceeded by an apostrophy(').
Press ↓, POSTAMBLE CODE: Press →, type exactly, **ESC,'x,'0 ENTER**.
Press ↓, BOLD ON: Type exactly, **ESC,'E,ESC,'G ENTER**.
Press ↓, BOLD OFF: Type exactly, **ESC,'F,ESC,'H ENTER**.
↓, UNDERLINE ON: Type exactly, **ESC,'-,'1 ENTER**.
↓, UNDERLINE OFF: Type exactly, **ESC,'-,'0 ENTER**.
↓, SUBSCRIPT ON: Type exactly, **ESC,'S,'1 ENTER**.
↓, SUBSCRIPT OFF: Type exactly, **ESC,'T ENTER**.
↓, SUPERSCRIPT ON: Type exactly, **ESC,'S,'0 ENTER**.
↓, SUPERSCRIPT OFF: Type exactly, **ESC,'T ENTER**.
↓, TRANSLATE1: Type exactly, **'£ ENTER**.
Do not change any lines that are not required by these instructions to be changed.
Preamble Code line nemonics initialize the printer, clear the printer buffer and, turn on the Near Letter Quality mode, which results in the N.L.Q. indicator light on the control panel of the QL printer turning on.
Postamble Code line nemonics turn off the Near Letter Quality mode after the document is through printing.

Bold On line nemonics turn on both the Bold type mode and, the Double strike type mode. Both modes are used to provide a high level of character enhancement over the Near Letter Quality character mode.
The only other line of intrest is TRANSLATE1. Its sole character is the pound (£) symbol. This allows Quill to print a £ symbol on the Sinclair QL Printer.
After you have the codes set, press F5. This will install and save the printer driver on your Quill Backup. Back up this cartridge. To install QL-NLQ on Abacus; before escaping the printer list, simply remove the Quill Backup from mdv1_ and, place the Abacus backup cartridge in mdv1_. Be sure the cursor is on QL-NLQ press **F5**. This will install and save the printer on the Abacus backup cartridge. To install the printer driver on Archive, reset the QL and, enter either the monitor or, TV modes by pressing F1 or, F2. Place the Archive Backup cartridge in mdv2_ and, type; **delete mdv2_printer_dat, Enter**. Place Quill-NLQ or, Abacus-NLQ cartridge in mdv1_ and, type; **copy mdv1_printer_dat to mdv2_printer_dat, Enter**. This will install the NLQ printer driver on the Archive cartridge. Backup these cartridges also. @ is shift 2.

DRIVER NAME	:QL-NLQ
PORT	:ser1
BAUD RATE	:9600
PARITY	:None
LINES/PAGE	:66
CHARACTERS/LINE	:80
CONTINOUS FORMS	:YES
END OF LINE CODE	:CR,LF
PREAMBLE CODE	:ESC,@,CAN,ESC,x,1
POSTAMBLE CODE	:ESC,x,0
BOLD ON	:ESC,E,ESC,G
BOLD OFF	:ESC,F,ESC,H
UNDERLINE ON	:ESC,-,1
UNDERLINE OFF	:ESC,-,0
SUBSCRIPT ON	:ESC,S,1
SUBSCRIPT OFF	:ESC,T
SUPERSCRIPT ON	:ESC,S,0
SUPERSCRIPT OFF	:ESC,T

TRANSLATE1	:£
TRANSLATE2	:NONE
TRANSLATE3	:NONE
TRANSLATE4	:NONE
TRANSLATE5	:NONE
TRANSLATE6	:NONE
TRANSLATE7	:NONE
TRANSLATE8	:NONE
TRANSLATE9	:NONE
TRANSLATE10	:NONE



TEXT ADVENTURE GAME EXAMPLE FOR THE SINCLAIR QL

by Dave and Rob Johnson

Here is a TEXT ADVENTURE GAME for the Sinclair QL called TODDL-QUEST. It is intended to show other programmers how a basic text adventure game is constructed. As it is shown, adventures are not hard to make, but a "mapping-out" of the setting and the use of a flow chart is also recommended.

```
10 WINDOW 512,256,0,0:CLS:PRINT:CSIZE 3,1:PRINT" TODDLEquest",,," by Dave & Rob
Johnson",," for the Sinclair QL 1987":PAUSE 50:CSIZE 0,0:a=0
20 PAUSE 150
100 PRINT"Background: You are five years old, and you are trying to get into you
r house."," The problem is the door knob is to high to reach!"
105 RESTORE
110 PRINT:PRINT:commands:READ word$
120 PAUSE 350
130 PRINT:PRINT"You are at the door, the knob looms above you.","Near you is a "
;word$:PRINT"What will you do?":INPUT a$
140 DATA "CHAIR","ROCK","DOG"
145 IF a$="up" THEN PRINT"Your too short to do that!....":GO TO 120
150 IF a$="dn" THEN PRINT"Where are you going, to dig a  hole?...":GO TO 120
155 IF a$="north" THEN PRINT"That's better now your going some where...":GO TO 3
00
160 IF a$="south" THEN GO TO 300
165 IF a$="east" THEN GO TO 300
170 IF a$="west" THEN GO TO 200
200 PAUSE 300:a=a+10:PRINT"You see before you a window, what will you do?  ( OPE
N / LOOK )":INPUT b$
210 IF word$="CHAIR" AND b$="open" THEN PRINT"You take the ";word$;" and open th
e window.":GO TO 3000
215 IF word$<>"CHAIR" AND b$="open" THEN PRINT"You loose your balance and fall a
nd hit your head....":PAUSE 350:death
220 IF b$="look" THEN PRINT"Your too short to do that.....":GO TO 110
230 STOP
300 IF a$="south" THEN GO TO 1000
310 IF a$="east" THEN GO TO 1010
330 IF a$="north" THEN GO TO 1000
340 IF a$="west" THEN GO TO 1000
500 PRINT "YOUR OPTIONS ARE: FORWARD OR RETURN ( to the door. )": INPUT a$
510 IF a$="forward" THEN PRINT"The Ice-cream truck turns around as you walk down
 the street."," You can see it weave back and forth. Now you realize why there i
s a red spot on"," the front of the truck, as it hits you!":PAUSE 350:death
520 IF a$="return" THEN GO TO 120
1000 PRINT" You are, or seem to be, trying to leave home and the anti run-away b
racelet will"," call the police and you will start life with a crimenal record t
hus unable to"," continue play. ":PAUSE 500:INPUT a$:IF a$="south": ELSE GO TO 1
20
1010 PRINT " Toddleing along the sun is in your eyes and you can see the ice-cre
am man weaving"," through traffic with a red spot on the front fender.  Quickly
you dig into your"," pocket  and find the quarter that you have been saveing, yo
u wave it at"," the man in the ice-cream truck but he only looks at you with  a
drooling smile"," as he races passed without stoping. Well you are at the corner
 of your home,"," which way will you go?":GO TO 500
2000 DEFine PROCedure death
2010 CLS:CSIZE 3,1:PRINT:PRINT"** YOU ARE DEAD!! **"
2020 PRINT"HA! HA!! HA!!!":PRINT"YOU HAVE COMPLETED: ";a;"% OF THE   ADVENTURE."
2021 BEEP 0,1,1300,150,1
2022 FOR i = 1 TO 150:NEXT i:BEEP
2025 STOP
2030 END DEFine
2040 DEFine PROCedure commands
2045 PRINT"********************"
2050 PRINT"** COMMANDS ARE ***"
2055 PRINT"********************"
2060 PRINT"UP, DN, NORTH, SOUTH, EAST, WEST, FORWARD, RETURN, LOOK"
2065 END DEFine
```

```
3000 PRINT"The window is open and you fall in. Before you, you see the living ro
om. You are very hungry and you see a mouse beside your Dad's Sinclair QL and yo
u eat it! YEECH!!! What will you do?  ( LOOK / RETURN )":a=a+10:INPUT b$
3010 IF b$="look" THEN PRINT"To your left is a doorway and to your right is a wi
ndow. There is nothing","of intrest here."
3020 IF b$="return" THEN GO TO 200
3030 a=a+10:PRINT"You enter the living room and your Dad is sitting in his chair
. He gets up from his","chair and gives you a spanking because you were outside
and you know your not sup-"," posed to be out there in the first place.":PAUSE 3
50:win
3050 DEFine PROCedure win
3055 CLS:CSIZE 3,1:PRINT:PRINT"    YOU WIN!!!":PRINT"YOU HAVE COMPLETED":PRINT a;
"% OF THIS ADVENTURE."
3060 END DEFine
```

MULTI-QUILL COPIER

by Joe Newman

Have you ever wanted to make more than one printout of a QUILL file? Have you ever tried making more than one printout? If so, you know what a task it is...a hassle. In order to print a file, you must hit F3, then P, then hit ENTER three times. And that only prints one copy of the file. You must wait for the file to print, then repeat the above procedure for each copy. Assuming you can't just run to a photocopier every time you need multiple copies of a Quill file, the following program is for you.

With my "Multi-Quill" copier program, you can make as many copies of a Quill program as you would like, with ease, and even unattended! Here's how:

Type in the copier program and save it. You can shorten the prompts and leave out the REM lines to help save typing time. I have tried to make the program easy for all, regardless of programming experience. The program is device independant, so it will work with disk or microdrive. If you have another type of mass storage, you just have to change the drive specifications in line 60.

Now run Quill and either type or load in a file. Then, instead of printing the file to the printer, you will print it to your storage device. To do this, hit F3, P, then ENTER only TWICE. Now, instead of "to printer", type in a name for the file (i.e., REPORT). Then hit ENTER. The file will now be saved in the exact same format as if it were being printed. You can use the name a file already has, as files printed to storage are saved with the LIS extension, NOT DOC.

You may now reset your QL and LRUN the copier program. Make sure you disk or tape with the file you want to print is in a drive. The program will ask you for the drive number you wish to use. You can enter 1 or 2 (or higher if you have extra drives). You will then be asked if you are using disks...enter Y or N. Then enter the name of the file you wish to printout...do NOT include the drive specification or extension (i.e., just REPORT). Now you will be asked to confirm your choices. When they are correct, you will be asked how many copies you wish to print. Once you enter a number (MAKE SURE YOUR PRINTER IS ON AND PAPER IS LINED UP CORRECTLY!) printing will commence. You can now kick back as the program handles all the printing.

Here is how it all works:

Line 70 combines your drive specification with your chosen drive number.

Line 135 then combines this with your file name and the _LIS extension (i.e., MDV1_REPORT_LIS).

Lines 140 to 160 are the loop which controls the printing. Your file is just copied right from storage to SER1 port. Since your file was printed to tape/disk, when it is copied to the printer it will print correctly, including any embedded control codes.

Line 155 prints a space to the printer just to make sure the printer buffer completely empties for each page. The CHR$(12) is the Epson (QL Printer) code for a form feed. This makes sure that each copy of the file is on a new page. You can remove this if blank pages are left between each copy.

I hope you put this program to good use. If you have any questions or comments, call or write to me: Joe Newman, Variety Sales, 325 W. Jersey St., #2D, Elizabeth, NJ 07202.

```
10 REMark *** MULTI-QUILL COPIER
20 REMark ***+++ by JOE NEWMAN +++
30 REMark     Summer, 1986
40 CLS: CLS #0: CLS #2:INPUT #0, "Which drive is your file on? (1-2)";drive
50 INPUT #0,"Are you using disks? (Y/N)";ans$
60 IF ans$="Y" OR ans$="y" THEN d$="FLP": ELSE d$="MDV"
70 drive$=d$&drive&"_"
80 INPUT #0,"Name of file (without extension)=";file$
90 CLS #0
100 PRINT "Your choices=":PRINT "Drive=";drive$:PRINT "File=";file$
110 INPUT #0,"All correct? (Y/N)";ans$:IF ans$="n" OR ans$="N" THEN RUN
120 INPUT #0,"Number of copies to make?";copies
130 CLS: CLS #0
135 copy_doc$=drive$&file$&"_LIS"
140 FOR loop=1 TO copies
150 COPY copy_doc$ TO ser
155 OPEN #3,ser1:PRINT #3," ":PRINT #3,CHR$(12):CLOSE #3
160 NEXT loop
```

The Heart of the Sinclair QL...the 68008

by Real Gagnon
(courtesy of SUM Magazine)

There is great confusion about the QL. The question is: in which computer category does the QL fall? The 32, the 16 or the 8-bit computer!?! It depends on the point of view. For now, we can say that the microprocessor, the MC68008, which is the heart of the QL, is an 8/32-bit type.

The microprocessor or the Central Processing Unit (CPU) is the nerve center of all activities in the computer system. It has generally 3 major tasks:

1-Control of operations
2-Interpretation of data
3-Execution of instruction

But a CPU alone is not very useful. The CPU needs memory and some peripherals to communicate with the outside world. The communication is possible via a computer BUS made up of multiple connectors which carry information to or from the CPU. A computer has generally three main bus types:

1-Address bus
2-Data bus
3-Control bus

Until now, most home computers were built around an 8-bit CPU. (8-bit CPUs are designed to work on 8-bit words at a time.) The two most popular of these were the Z80 (ZX81, TS2068, TANDY) and the 6502 (C64, ATARI). The 68008 of the QL is, like we previously called, an 8/32-bit because it has an 8-bit wide data bus and works internally on 32-bit words. On this point, the 68008 is similar to the INTEL 8088 (IBM PC) which is an 8/16-bit CPU because of its 8-bit data bus and 16-bit internal structure. Apple's Macintosh is a 16/32-bit with its MC68000.

MC68008 vs MC68000: THE HARDWARE SIDE

The MC68008 is the "economical version" of the MC68000. In fact, the MOTOROLA's 68000 series forms a real family. The 68000, itself, has a 16-bit wide data bus, the 68008 has an 8-bit one and the 68020 has the 32-bit wide data bus. The "advantage" of the 68008 over the other members of the family is, like we said, economical because the most commonly available low cost support chips are organised on an 8-bit basis. This means that these can be more easily designed into a system around the MC68008. Also the necessary connectors are smaller on 8-bit data bus systems than on a 16-bit data bus systems.

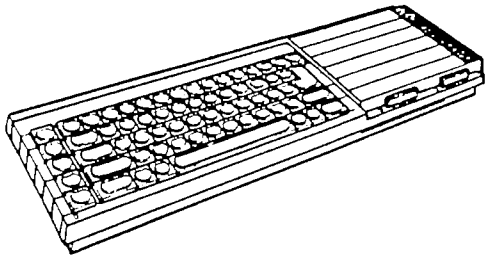
The address bus has been reduced on the 68008 to 20 address lines (24 on the 68000) and 3 of the 68000's control signals are not directly present on the 68008. These omissions are justified by the fact that the 68000 is a 64-pin I.C. and the 68008 is a 48-pin I.C.. With 20 address lines, the QL can address directly 1MB of memory. There is no need here for a complicated BANK SWITCHING system like on the COMMODORE 128 and ATARI 130XE. On the QL, all the memory is accessible anytime. The 68000 has 2 pins not present on the 68008, UDS (upper data strobe) and LDS (low data strobe), but on the 68008 the pin DS (data strobe) does the same job. Also the VMA (Valid Memory Address) is not present, an external circuit is required here to simulate it.

MC68008 vs MC68000: THE SOFTWARE SIDE

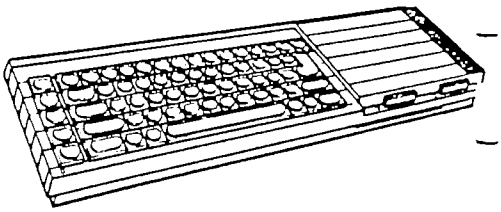
The 68008 has the SAME instruction set as the other members of the 68000 family. With only 56 instruction types and the 14 different addressing modes, the programmer can access up to 1000 powerful instructions. These instructions can be from 1 to 5 words long, that is 2 to 10 bytes long. The 68008 (like the 68000) has 17 all-purpose 32-bit wide registers, one 16-bit FLAGS register and a 32-bit Program Counter. The 32-bit all-purpose registers means SPEED because complex operations can be carried out internally to the 68008 with the minimum need of data to or from memory. Each one of the all-purpose registers can be used like an Accumulator and we can do operations on 1 bit, 4 bits (DIGIT), 8 bits (BYTE), 16 bits (WORD) and 32 bits (LONG WORD).

In conclusion, if you love machine code programming then the Sinclair QL is certainly a good choice.

Editor's Note: Mr. Gagnon is the editor of SIN, "the only French North American Sinclair oriented magazine".



MORE ON QOOLING THE QL



by John E Juergens

I would like to express my thanks to M. Vincent Lyon for his article Qool QL, TD V3, #5, Jul-Aug '87 issue.

Although I had not suffered a heat problem loss on our QL prior to his article the great amount of heat present on the top right side of the computer has always bothered me. I knew that sooner or later it would cause problems or failure.

I would have done something long before reading "Qool QL" but no idea of the merit Mr. Lyon supplied occurred to me.

I have implemented his suggestion about the Radio Shack 3" brushless DC fan with only slight modification. Unlike some suggestions, it works completely to my satisfaction.

As I do not care to have the fan's exhaust blowing on me and because a 9 VDC adaptor runs the fan faster than our needs, causing excess exhaust and noise, three additional things were done: Legs on the fan, the fan on top of the QL and powered with less voltage.

The legs and the placement of the fan allow the exhaust to be directed upward, away from myself and cooling to occur along the originally engineered lines, i.e. by "pulling" heat out of the upper right exit vents.

Although the Radio Shack specifications state that the fan requires 7-12 VDC, it seems to run well at 5.8 VDC providing sufficient cooling for our local temperatures and almost silent operation.

The fan has pre-drilled holes at each corner, front and back, for a total of 8. Two-2" and two-4", 4-40 bolts (or slightly larger) will fit the 4 back holes. Nuts, two per bolt, will allow leg height adjustment which in turn adjusts the angle of the fan and exhaust with respect to the QL. Four short lengths of rubber windshield wiper tubing may be cut and pushed over the bolt ends to further lessen noise and maintain fan placement.

The 5.8 VDC comes from a multi-voltage adaptor which I had purchased from BEST Co. sometime ago for $7.95 and which I regularly use to run a small printing calculator. Radio Shack also sells a multi-voltage adaptor (R/S 273-1650, $11.95, '88 Cat.)

My adaptor has a voltage switch which allows selection of 3, 4.5, 6, 7.5, 9 and 12 VDC and a current switch allowing a choice of 200 or 500 ma. The Radio Shack adaptor appears to have only a voltage switch allowing 3, 4.5, 6, 7.5 and 9 VDC at 300 ma. but should work as well.

After experimenting I found that the 4.5 VDC and 200 ma. settings worked best for me. They produce the measured 5.8 VDC @ 55 ma., adequate cooling, near-silent operation and still allow use of the calculator.

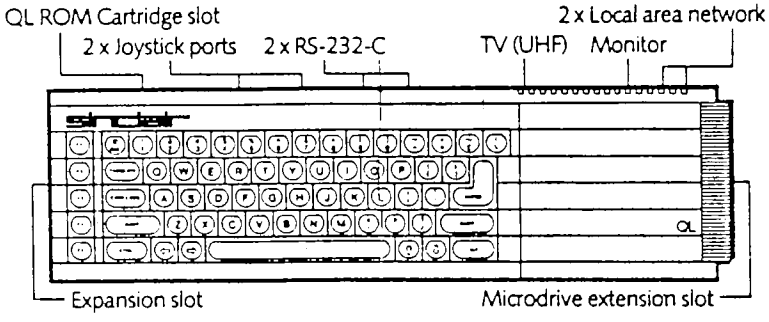
It should be kept in mind that some hooking up is necessary as the fan comes with two wires which needs be attached to the adaptor. The adaptor comes ready to plug into any number of things but not into the fan. A couple of bright spots here is that if one incorrectly hooks the adaptor to the fan, the fan won't work but no harm is done. Just reverse the leads. Even if one can't determine which lead is plus or minus, one has a 50% chance of getting it right the first time by simply guessing.

The fan is placed with the two rear legs on the upper right corner of the QL over the upper heat exit vents. The two front legs rest on whatever supports the computer. Although the QL power supply and monitor plugs occupy space right below the heat vents and seem to block fan placement the front legs are sufficiently wide apart to allow them to clear the plugs, offer a fairly wide adjustment range and do the job.

The adjustable power supply should just about guarantee a Quietly Qooled QL even on the hottest day no matter how long one operates it.

A word of caution may be needed: The fan appears to use a permanent magnet along with magnetic flux built up by the adaptor current to operate. Stray magnetism can alter/destroy taped data. Although the fan sits far enough away from the microdrives to pose no observed threat to cartridge data integrity it would not be prudent to put or pass cartridges any closer to the fan, whether running or not, than that required to insert/remove them from the microdrives. In short, keep cartridges away from the fan!

In balancing the rather small amount of money and effort involved against the satisfaction I now get from touching the previously very hot top right side of our QL and finding it completely and quietly cooled, the bottom line is all plusses and no negatives so far.



Why the QL?

by Mike de Sosa

1. The Sinclair QL is, quite simply, the best value for money of any professional computer

2. Second- and third-generation QL software--far cheaper and in many cases better than most big-name brands--is now becoming available

3. Many QL users who also use big name computers at work, prefer the QL for many reasons.

4. The QL Users and Tinkers Association (QUANTA) library now provide twenty quad-density floppy disks containing QL software of all types and descriptions at low-cost to members--contact Tom Bent for further information on QUANTA, its library, & its newsletter.

5. Low-cost and innovative QL firmware and hardware is only just now approaching fully developed status, utilizing the QL's full power

6. The T/S 2068 is an excellent computer, but it lacks the capacity to run comprehensive database, spreadsheet, desktop publishing, and integrated (multitasked) programs. The QL with TRUMP CARD now makes use of 896K of RAM.

7. Most QL users are unaware of or do not make use of the latest QL products which can transform operations of the QL to such an extent that it should be the computer of choice for many!

To take fuller advantage of your QL, subscribe to TIME DESIGNS Magazine ($16.95 a year for six issues) and purchase Dr. Mike de Sosa's excellent book, TAKING THE QUANTUM LEAP, the most up-to-date book on the QL, written in language anyone can understand, which explains many useful things about the QL not found in its user guide. Both are available from TIME DESIGNS, 29722 Holt Rd., Colton, OR 97017; tel. (503) 824-2658; CompuServe ID# 71350,3230. Order both now!